The Dissertation Committee for Doo Soon Kim
certifies that this is the approved version of the following dissertation:

Knowledge Integration in Machine Reading

Committee:

Bruce W. Porter, Supervisor

James F. Allen

Kenneth J. Barker

Vladimir Lifschitz

Raymond J. Mooney

Knowledge Integration in Machine Reading

by

Doo Soon Kim, B.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

August 2011

Acknowledgments

First and foremost, I would like to thank my advisor, Bruce Porter, for guiding me through the long journey of my PhD research. Beyond being an academic advisor, he has been my mentor and role model in everything. It is truly a blessing to have studied with him.

I would also like to thank the members of my committee, Ken Barker, Ray Mooney, James Allen, and Vladimir Lifschitz for their comments and suggestions. I am particularly grateful to Ken for his help on my research over the years.

I am indebted to several researchers, with whom I have worked through several projects. Among them, I especially want to thank Peter Clark and Christopher Welty.

I would also like to thank my colleagues in the Knowledge Representation and Reasoning group, James Fan, Peter Yeh, Dan Tecuci, Jason Chaw, and Michael Glass. My research has benefited greatly from discussions with them.

I would like to thank Lydia Griffith, Katherine Utz and the staff of the Department of Computer Sciences. I especially thank Stacy Miller for her administrative work that supported my research. She has also been a good friend, and I will miss having lunches with her.

Thank you to my friends at the Korean Baptist Church of Austin, especially my spiritual mentor, Pastor SunBae Kim.

This research would not have been possible without the dedicated support of my family. My wife, Sujin, and my kids, Joshua and Allison, have been my joy and have energized me every day. My parents-in-law, YoonCheol Hong and BokNyon Ha; my parents, SunYo Kim and JungSook An; and my sister, Doo-Iee Kim, have always been behind me to encourage and support me.

Finally, I want to thank God, my rock and strength, for all things he has done for me in the last seven years.

The research presented here was funded by DARPA, IBM, and the University of Texas at Austin. I gratefully acknowledge their support.

Knowledge Integration in Machine Reading

Publication No. _____

Doo Soon Kim, Ph.D.
The University of Texas at Austin, 2011

Supervisor: Bruce W. Porter

Machine reading is the artificial-intelligence task of automatically reading a corpus of texts and, from the contents, building a knowledge base that supports automated reasoning and question answering. Success at this task could fundamentally solve the knowledge acquisition bottleneck – the widely recognized problem that knowledge-based AI systems are difficult and expensive to build because of the difficulty of acquiring knowledge from authoritative sources and building useful knowledge bases. One challenge inherent in machine reading is knowledge integration – the task of correctly and coherently combining knowledge snippets extracted from texts. This dissertation shows that knowledge integration can be automated and that it can significantly improve the performance of machine reading.

We specifically focus on two contributions of knowledge integration. The first contribution is for improving the coherence of learned knowledge

bases to better support automated reasoning and question answering. Knowledge integration achieves this benefit by aligning knowledge snippets that contain overlapping content. The alignment is difficult because the snippets can use significantly different surface forms. In one common type of variation, two snippets might contain overlapping content that is expressed at different levels of granularity or detail. Our matcher can "see past" this difference to align knowledge snippets drawn from a single document, from multiple documents, or from a document and a background knowledge base.

The second contribution is for improving text interpretation. Our approach is to delay ambiguity resolution to enable a machine-reading system to maintain multiple candidate interpretations. This is useful because typically, as the system reads through texts, evidence accumulates to help the knowledge integration system resolve ambiguities correctly. To avoid a combinatorial explosion in the number of candidate interpretations, we propose the *packed representation* to compactly encode all the candidates. Also, we present an algorithm that prunes interpretations from the packed representation as evidence accumulates.

We evaluate our work by building and testing two prototype machine reading systems and measuring the quality of the knowledge bases they construct. The evaluation shows that our knowledge integration algorithms improve the cohesiveness of the knowledge bases, indicating their improved ability to support automated reasoning and question answering. The evaluation also shows that our approach to postponing ambiguity resolution improves the

system's accuracy at text interpretation.

Table of Contents

Acknowledgments **iv**

Abstract **vi**

List of Tables **xiv**

List of Figures **xv**

Chapter 1. Introduction **1**

1.1 The Goals of this Dissertation 3

1.2 Summary of Contributions of this Dissertation 5

1.3 Organization of the Dissertation 6

Chapter 2. Knowledge Integration in Machine Reading **7**

2.1 Our Machine Reading Task and its Comparison to the Other Types of Machine Reading 7

2.1.1 Comparison to IE 11

2.1.2 Comparison to Single-text NLU 13

2.2 Component Tasks in Our Machine Reading System 14

2.2.1 Syntactic Processing 15

2.2.2 Semantic Processing 17

2.2.2.1 Word Sense Disambiguation 18

2.2.2.2 Semantic Role Labeling 19

2.2.2.3 Temporal Relation Identification 22

2.2.3 Discourse-level Processing 24

2.2.3.1 Co-reference Resolution 24

2.2.3.2 Indirect Anaphora Resolution 26

2.2.3.3 Discourse Relation Identification 27

2.3 Knowledge Integration in Machine Reading 29

2.3.1 Task Definition 30
2.3.2 Importance of Knowledge Integration 32
2.3.2.1 Knowledge integration improves reasoning . . . 33
2.3.2.2 Knowledge integration improves language interpretation 33
2.3.2.3 Knowledge integration facilitates knowledge base management 35
2.3.3 Challenges in Knowledge Integration 35
2.3.3.1 Aligning representations of the same content . . 36
2.3.3.2 Inferring unspecified information 38
2.3.4 Knowledge Integration in Other Fields of AI 40
2.3.4.1 Research in knowledge-based systems 40
2.3.4.2 Research in discourse-level processing 42
2.3.5 Our Project Description 44
2.3.6 Summary 45

Chapter 3. Knowledge Integration: Combining Knowledge Snippets Coherently 46

3.1 Kleo 47
3.1.1 The NL Component 48
3.1.1.1 Parser 49
3.1.1.2 Semantic Interpreter 49
3.1.2 The KI Component 53
3.1.2.1 Sentence-to-Sentence Knowledge Integration . . 53
3.1.2.2 Text-to-Text Knowledge Integration 54
3.1.3 Knowledge Base 54
3.2 Our Approach 55
3.2.1 Graph Matcher 55
3.2.2 Sentence-to-Sentence Knowledge Integration 63
3.2.2.1 Stitch 63
3.2.2.2 Elaborate 64
3.2.3 Text-to-Text Knowledge Integration 66
3.3 Evaluation 68

3.3.1 Evaluation of SKI 69
3.3.2 Failure Analysis on SKI 72
3.3.3 Evaluation of TKI 73
3.4 Related Work 77
3.4.1 Handling Granularity in Reasoning 78
3.4.2 Semantic Decomposition 79
3.4.3 Analogical Reasoning 80
3.4.4 Paraphrase Discovery 81
3.4.5 Underspecification 81

Chapter 4. Application of Knowledge Integration to Text Interpretation 84
4.1 Using Redundancy across Multiple Texts 88
4.1.1 Packed Representation 90
4.1.2 Combining Packed Representations 93
4.1.3 Ally: Prototype System 96
4.1.3.1 Parser 96
4.1.3.2 Semantic interpreter 97
4.1.3.3 Knowledge integration 98
4.1.4 Experiment 1 99
4.1.5 Experiment 2 103
4.1.6 Experiment 3 103
4.1.7 Summary 105
4.2 Using OntoNotes 105
4.2.1 OntoNotes 107
4.2.2 Packed Representation 108
4.2.3 Ally: A Prototype Language Interpreter 112
4.2.3.1 Producing syntactic packed representation . . . 113
4.2.3.2 Assigning word senses and semantic relations . 118
4.2.3.3 Knowledge integration 119
4.2.4 Experiment 1 121
4.2.5 Experiment 2 124
4.2.6 Summary 126

4.3 Using Prismatic . . . 127
4.3.1 Prismatic . . . 128
4.3.2 Our Approach . . . 130
4.3.2.1 Scoring candidate parse fragments . . . 131
4.3.2.2 Scoring candidate semantic types . . . 132
4.3.2.3 Scoring candidate semantic relations . . . 133
4.3.2.4 Extracting the overall best semantic representation 134
4.3.3 Experiment . . . 135
4.3.4 Discussion . . . 136
4.3.5 Summary . . . 140
4.4 Related Work . . . 141
4.4.1 Packed Representation . . . 141
4.4.2 Underspecified Representation . . . 142
4.4.3 Disambiguating Packed Representations . . . 143
4.4.4 System Architectures . . . 143
4.5 Summary . . . 144

Chapter 5. Future Work **146**
5.1 Aligning Semantic Representations . . . 146
5.1.1 Resolving Granularity Mismatch . . . 147
5.1.2 Viewpoint Difference . . . 148
5.1.3 Distinguishing Different Contexts . . . 149
5.2 Inferring Unspecified Information . . . 151
5.2.1 Adapting Background Knowledge Representation . . . 151
5.2.2 Reasoning for Inference . . . 153
5.3 Other Sources of Evidence for Text Interpretation . . . 153
5.3.1 Coherence of Paragraphs . . . 153
5.3.2 Reading Preliminary Texts . . . 154
5.3.3 Other External Knowledge Resources . . . 154
5.4 Other Future Projects . . . 155
5.4.1 Belief Management . . . 155
5.4.2 Content Selection . . . 156

5.4.3 Other Types of Texts 156
5.5 Applications and Wide-Scale Experiment 157
5.5.1 Competitive Intelligence 157
5.5.2 Social Analysis 158
5.5.3 Information Management 158
5.5.4 Application-Oriented Evaluation 158

Chapter 6. Conclusion **159**
6.1 Goals Revisited and Summary of Our Contribution 159
6.1.1 Coherently Combining Knowledge Snippets 159
6.1.2 Applying Knowledge Integration to Text Interpretation 161
6.2 Lessons Learned 162
6.3 Closing Words 164

Appendices **165**

Appendix A. Ten paraphrases of the heart text **166**

Appendix B. Converting packed representation to Alchemy statements **169**

Bibliography **172**

Index **198**

Vita **199**

List of Tables

2.1 Comparison of our task to the two major types of the machine reading tasks 12

2.2 The Component Library semantic roles used in our system . . 22

2.3 Four discourse relations. S1 and S2 are the two segments. . . . 28

3.1 The performance of the NL component in Kleo measured for 25 texts in each domain of the heart and the engine. 52

3.2 Types of granularity differences with examples: Granularity differences are shown in *italics*. The top sentence in the examples is fine-grained, the bottom coarse-grained. 56

3.3 The frequency of invocation of the extension pattern rules . . 77

3.4 The factors in the qualia structure and an example of the qualia structure for *scissor*. 80

4.1 Percentage of nodes and edges maintaining the correct types and semantic relations in the baseline system and Ally for all 37 sentences. 101

4.2 Evaluation Result 123

4.3 The upper bounds on Ally's performance. The ratio of the nodes containing the correct type (WSD) and the edges containing the correct relation (SR) in the packed representation . 124

4.4 Precision and recall for type assignment and semantic relation assignment. The baseline system is based on the traditional pipeline approach. Ally delays resolving ambiguities to jointly resolve them at the last step. Ally+Prism uses Prismatic for disambiguating the packed representation. 138

List of Figures

2.1 Three example texts about the human heart 9

2.2 Knowledge base constructed from processing the texts in Figure 2.1. The dark nodes with the thick borders are root nodes. 10

2.3 Two parse trees for *"She drinks water"*: (a) dependency parse (b) constituency parse 15

2.4 The fragmented parse trees 16

2.5 (a) a fragmented knowledge base and (b) an integrated knowledge base 34

2.6 Two knowledge integration operations 36

2.7 **(a)** Semantic representations for (1) and (4), which represent the blood circulation with different granularity; **(b)** Simple graph matching: `Move-1a2` has two destinations (incorrect); **(c)** Flexible matching: the subevent relations are established from `Move-2` to `Move-1a` and `Move-1b` (correct) 39

3.1 Architecture of Kleo: The arrows represent data flow. 48

3.2 The dependency parse for S1 produced by the Stanford Parser 49

3.3 The semantic representations for S1 and S2 50

3.4 The intermediate steps of converting the dependency parse for S1 to the semantic representation. (a) `engine-2` and `gasoline-7` are assigned a type (b) `piston-4` is assigned a type and its dependency relation, *poss* is converted into *is-part-of* (c) `compresses-5` is assigned type and its dependency relations, *nsubj* and *dobj*, are resolved 51

3.5 The outcome representation from combining S1 and S2 in Figure 3.3 54

3.6 The stitched representation 63

3.7 The example knowledge base 65

3.8 K-units produced from the representation in Figure 3.5 by partitioning. The dotted nodes are roots. 66

3.9 The average density-improvement over 25 texts in each domain. The experiment was repeated 10 times with the different order of reading. The lines represent the standard deviation. The difference between Mobius and Mobius+GM/Mobius+KB and between Mobius+GM/Mobius+KB is statistically significant in both domains ($p<.05$ for the two tail z-test). 71

3.10 Increase in knowledge-base size with presentation of triples to TKI 75

3.11 With partitioning, the run time of updating a knowledge base is almost negligible 76

4.1 (a) The traditional pipeline approach; (b) the beam approach to maintain multiple candidate interpretations 85

4.2 Our system architecture based on the packed representation . 86

4.3 Two candidate interpretations for each of S1 and S2. The interpretations, A2 and B2, which are identical, are more likely to be correct than the others because the sentences express the same meaning. 89

4.4 The packed representation for S1 (PR1) 91

4.5 Packed representation for S2, *"The engine's spark plug combusts gasoline."* 93

4.6 Syntactic packed representation for S1, capturing the prepositional phrase attachment ambiguity of *"with its spark plug"*. . 97

4.7 Correctness scores for Ally vs. baseline system on (a) type triples (type assignment task), (b) content triples (semantic relations assignment task) and (c) all triples (with the bars representing standard deviation). X-axis represents the values of N (from 1 to 37) and Y-axis represents the ratio of the correct triples. 102

4.8 Total number of triples in individual sentence packed representations (top); total number of triples in the packed representation after merging in Ally (middle); total number of triples after pruning to the highest scoring alternative (bottom). . . . 104

4.9 Sensitivity of Ally and the baseline system to the quality of the NL system output. The quality of the triples produced by the NL component is perturbed, affecting performance accuracy of the two systems. For example, when the quality of the NL output is perturbed to the level of 70% accuracy, Ally achieves a higher accuracy, 80%, using Algorithm 2. The arrow indicates unperturbed language interpreter performance. 104

4.10 Example of the word sense and semantic role annotations in OntoNotes. The sense annotation, `1 plant-n 1`, represents that *plant* (index: 1) is a noun (plant-n) and that its sense choice is the first sense. The semantic role annotation (last line) represents that *change* (index: 6) is a verb (change-v), maps to the frame, `change.01`, and has two roles: ARG0 (*"The plant and another next door"*) and ARG1(*"the face of Postvilles"*). 0:2 represents the word span covered by a non-terminal in the constituency parse tree – the grandfather (indicated by 2) of the first word (indicated by 0). 108

4.11 A packed representation 110

4.12 The architecture of Ally 113

4.13 Three candidate parses. To save space, some triples common to all three parses are omitted. The bold font indicates the differences among the parses. 114

4.14 Syntactic packed representation for S1 114

4.15 The result of step1 115

4.16 The result of step2 115

4.17 Compression ratio as the number of the alternative parses varies 125

4.18 Compression ratio 126

4.19 Parse tree produced by the ESG parser [100] for *"In 1921, Einstein received the Nobel Prize for his original work on the photoelectric effect"*. 129

4.20 Prismatic Frames that are produced from the parse tree in Figure 4.19. (a) is produced by applying S-V-O to the parse tree. (b) is produced by replacing *Einstein* in (a) with PERSON. (c) is produced by applying S-V-P-O to the parse tree. 130

4.21 Semantic frame about *"rush (in the football game)"* 138

5.1 (a) The one chamber pumping model; (b) The semantic representation that describes the heart with the two chambers . . . 151

5.2 The two chamber pumping model adapted from the one chamber pumping model 152

Chapter 1

Introduction

Artificial Intelligence (AI) reasoning technologies can vastly improve human productivity of life by automatically performing intelligent tasks. For example, a medical diagnostic system might allow laymen to answer questions and arrive at a diagnosis without the need to consult medical experts. A decision support system can assist people or organizations by guiding them through questions and considerations pertinent to their decision process.

These reasoning systems commonly require one crucial element, a vast amount of background knowledge – information the system should know to make inference. For example, consider a medical diagnostic system that infers the disease given the observed symptoms. To perform such inferences, the system needs various types of knowledge, including factual knowledge (e.g., the definitions and the characteristics of symptoms and diseases, the anatomical structure of the human body and so on) and procedural knowledge (e.g., analysis and diagnosis procedures applied to the symptoms and lab results to suggest candidate diseases).

Unfortunately, it is difficult and time-consuming to build a knowledge base that can supply this kind of background knowledge. First, it is challeng-

ing to acquire knowledge from domain experts. Knowledge elicitation from them often requires extensive training because of the difficulty of verbalizing background knowledge, which is often unconsciously used. Furthermore, it is non-trivial to determine in advance which elicited knowledge should be contained in the background knowledge base. Finally, it is challenging to formally represent the acquired knowledge. This formal encoding, usually performed by skilled knowledge engineers, requires high expertise and, thus, is expensive. [1] All these problems could be compounded if the dynamic information in the knowledge base changes frequently.

Computers could address this knowledge acquisition bottleneck by reading texts, extracting information, and building an inference-capable knowledge base on their own. This approach, without relying on manual human effort, is promising because a vast amount of human knowledge is contained in texts. For example, news articles, encyclopedias, and Wikipedia [2] provide a vast amount of world knowledge, and the blogs and the micro-texts in social media provide expert opinions about a variety of subject matter. Moreover, the amount of textual information available increases rapidly. The recent progress of Natural Language Processing (NLP) reinforces the promise of this approach with the improved capability of extracting content from texts.

If a machine reading system can successfully produce a formal knowl-

[1] In Project Halo [54], the cost for a team of knowledge engineers to encode 50 pages was $10K ($200 per page).

[2] `http://www.wikipedia.org`

edge base, it will be pivotal in deploying reasoning technologies across many domains, supplying background knowledge at a very low cost.

1.1 The Goals of this Dissertation

This dissertation addresses one important task in machine reading: knowledge integration, the task of combining knowledge snippets into a coherent whole. When NLP software generates the semantic representations of sentences or phrases, these representations (knowledge snippets) should be combined along with the prior knowledge the system possesses to produce a single coherent knowledge base.

This dissertation particularly focuses on two hypotheses about knowledge integration.

Hypothesis 1. *Knowledge integration can improve the reasoning performance of the output knowledge base.*

Our first goal is to show that coherently combining knowledge snippets is critical to knowledge integration because a fragmented knowledge base (in which knowledge snippets are unrelated) or an incorrectly combined knowledge base are both unsuitable for use by a reasoning system. The fragmented knowledge base would fail to deliver a coherent set of relevant pieces of knowledge. Knowledge integration, however, is challenging because knowledge snippets can be combined in many different ways, and identifying the correct combination is difficult.

Ultimately, knowledge integration's contribution to the reasoning systems should be evaluated by measuring the performance gain in reasoning tasks, but this application-oriented extrinsic evaluation is non-trivial. Instead, we evaluate the constructed knowledge base intrinsically in terms of two metrics: the cohesiveness [3] and the correctness of the knowledge base.

Hypothesis 2. *Knowledge integration can improve the accuracy of text interpretation.*

Our second goal is to show that knowledge integration could benefit various tasks in text interpretation, such as parsing, word sense disambiguation, and semantic role labeling. In the interpretation of one text, other texts or knowledge bases may exist, which contain information useful for the interpretation. Knowledge integration can improve the accuracy of an interpretation because it can access multiple texts and knowledge bases simultaneously. A key challenge in this approach is to design an architecture in which knowledge integration interacts with language interpretation in a bi-directional manner: knowledge integration receives semantic representations from language interpretation, while language interpretation receives useful evidence from knowledge integration.

[3] Cohesiveness can approximate the coherence of the knowledge base because a coherent knowledge base is usually cohesive.

1.2 Summary of Contributions of this Dissertation

In this dissertation, we present the evidence that supports Hypotheses 1 and 2.

For Hypothesis 1 (as explained in Chapter 3), we developed a proof-of-concept machine reading system, Kleo, which is equipped with knowledge integration facilities. In particularly, Kleo addresses granularity mismatches among knowledge snippets and uses the contents it learned from reading previous texts to help integrate the knowledge snippets. An evaluation of our knowledge integration methods shows that they improve the cohesiveness of the knowledge base without degrading its correctness.

For Hypothesis 2 (as explained in Chapter 4), we developed an architecture in which the system can maintain multiple candidate interpretations of a text until knowledge integration acquires strong evidence to choose one of the interpretations. In particular, we developed a representation scheme called the *packed representation* to efficiently manage a myriad of candidates. We also explored three sources of evidence that knowledge integration could exploit for selecting a correct interpretation. These sources are: redundancy across multiple texts, OntoNotes (a semantically annotated corpus) [69], and Prismatic (a knowledge base automatically constructed from texts) [48]. For redundancy and OntoNotes, our evaluation shows that knowledge integration can significantly improve the accuracy of text interpretation by using packed representation.

1.3 Organization of the Dissertation

This dissertation is organized as follows.

Chapter 1 introduces machine reading as a solution to the Knowledge Acquisition Bottleneck and presents the overview of this dissertation.

Chapter 2 presents the architecture of our machine reading system and explains its NLP components. The chapter, then, introduces our knowledge integration task, which has received little attention in the NLP research.

Chapter 3 presents our project that evaluates Hypothesis 1 and introduces a prototype machine reading system, Kleo, and its knowledge integration facilities. The chapter also presents the positive evaluation results of our approach, which significantly improves the quality of the output knowledge base.

Chapter 4 presents our project that evaluates Hypothesis 2 and presents packed representation, which allows the system to delay ambiguity resolution, and the knowledge integration algorithms that resolve the ambiguities in the packed representation. The chapter also presents the evaluation results, which show that our approach significantly improves the quality of the semantic representations.

Chapter 5 presents our future work plans on knowledge integration.

Chapter 6 concludes with a summary of this dissertation.

Chapter 2

Knowledge Integration in Machine Reading

Machine Reading is a general term used to describe various types of reading tasks, from Information Extraction (so-called macro reading) [39] to deep analysis of texts (micro reading) [128]. Section 2.1 defines our reading task precisely and compares it to other types of machine reading. Then, we discuss various components that are needed to build our machine reading system. Specifically, in Section 2.2, we review the Natural Language Processing (NLP) components that have been extensively studied. These components alone are, however, insufficient to build our system, requiring sophisticated knowledge integration. Section 2.3 introduces our knowledge integration task in detail.

2.1 Our Machine Reading Task and its Comparison to the Other Types of Machine Reading

It is our goal to build a multi-text reading system that can build formal representations of the content in the texts to enable automated reasoning. In particular, the system will read full-fledged English texts (in contrast to simplified English [33]) that describe conceptual knowledge to build an inference-capable formal knowledge base. Figure 2.2 shows an example of the kind of

knowledge base we want to build, given the three texts shown in Figure 2.1.

A primary component of the knowledge base is its graphical representation, a formal language with well-defined semantics. The nodes (see Figure 2.2) represent the semantic concepts, and the edges represent the semantic relationship between two nodes. These semantic concepts and relations are defined in a formal ontology. In our project, we use the Component Library [11] as our ontology, which provides approximately 800 domain-independent concepts and 80 semantic relations such as temporal, spatial, meronymic, and causal relations.

Quantification can also be represented in the graphical representation – the root node is a universally quantified concept, and the other nodes are existentially quantified in the scope of the root node. For example, the top-left representation in Figure 2.2 logically represents $\forall x.Heart(x) \rightarrow \exists yz....[Pump(y) \wedge subclasses(x, y) \wedge Valve(z) \wedge haspart(x, z)...]$. This graphical representation has been widely used in the knowledge-based systems [10] [54], providing the expressiveness and the tractability in reasoning.

We now compare our task (multi-text machine reading) to two major classes of machine reading: Information Extraction (IE) and a single-text Natural Language Understanding (NLU). Table 2.1 summarizes their comparison. Two major criteria characterize the reading tasks: the target representational language and the properties of the corpus. The target representational language could vary from simple English phrases to complex forms of logical representations. As the representational language becomes more expressive, it

text1: "Hearts are valved, muscular pumps that propel blood around the body. Hearts consist of one or more muscular chambers connected in series and guarded by valves or, in a few cases, sphincters (e.g., in some molluscan hearts), which allow blood to flow in only one direction. The mammalian heart has four chambers: two atria and two ventricles. Contractions of the heart result in the ejection of blood into the circulatory system. Multiple heart chambers permit stepwise increases in pressure as blood passes from the venous to the arterial side of the circulation."

text2: "Hearts pump blood through the body. Blood carries oxygen to organs throughout the body. Blood leaves the heart, then goes to the lungs where it is oxygenated. The oxygen given to the blood by the lungs is then burned by organs throughout the body. Eventually the blood returns to the heart, depleted of oxygen. It is then pumped by the heart back to the lungs."

text3: "This is a subject that is near and dear to my heart. The heart is a two sided, four chambered pump. It is made up mostly of muscle. Heart muscle is very special. Unlike all the other muscles in the body, the heart muscle cannot afford to get tired. Imagine what would happen if every 15 minutes or so the pump got tired and decided to take a little nap! Not a pretty sight. So, heart muscle is always expanding and contracting, usually at between 60 and 100 beats per minute."

Figure 2.1: Three example texts about the human heart

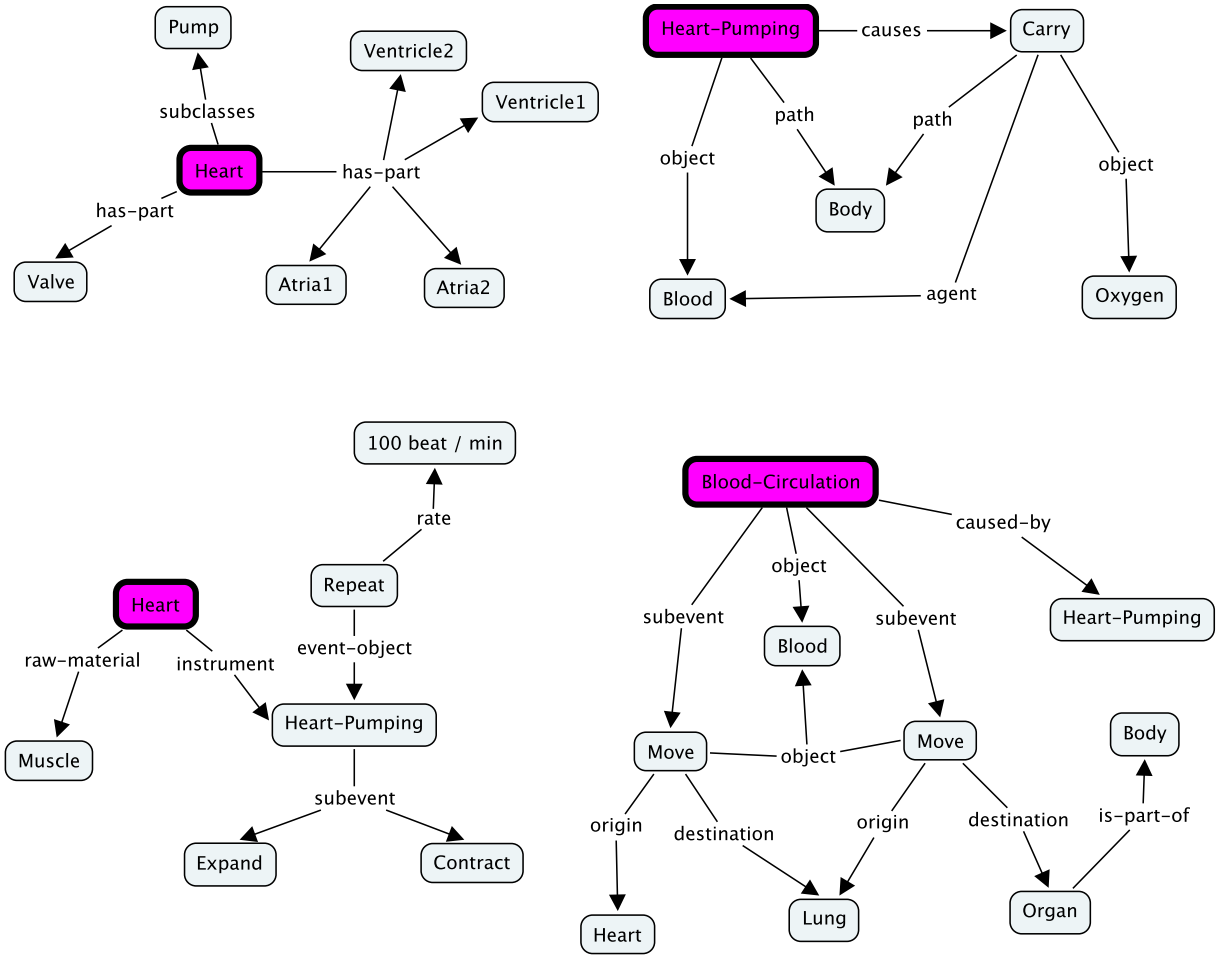


Figure 2.2: Knowledge base constructed from processing the texts in Figure 2.1. The dark nodes with the thick borders are root nodes.

requires more sophisticated language processing. Another important factor for characterizing the machine reading task is the properties of the corpus itself. The system could read only one text, focusing on a very specific topic, or a large corpus of the texts (possibly at the web-scale), which includes various genres and domains.

2.1.1 Comparison to IE

The main difference between IE and our task relates to the kinds of knowledge they extract: IE generally extracts pre-defined (usually simple) types of knowledge, such as named entities (e.g., cities or persons) [43] or the relations between them [7], whereas our task is to build knowledge representations with a more expressive language (graphical representation). Therefore, our task can represent more and various types of knowledge such as the description of events, the relations between events (such as causal or temporal relations), and quantification.

The IE task involves extracting knowledge that is explicit in a text, i.e., the knowledge that is available on the surface of the text. In contrast, our task involves extracting knowledge that might only be implicit in a text. Furthermore, in IE, an ontology is minimally used to ground the extracted information, whereas our task grounds the semantic representations in the formal ontology. Because the IE task attempts to extract only limited types of knowledge, IE generally applies shallow NLP techniques, such as chunking, rather than full NLP required to understand a whole text. The advantage of

	IE	**Our task (multi text machine reading)**	**Single text NLU**
Expressive power of the representation language	Named entities or relations between the two named entities	Limited first-order logic	First-order logic with several extensions
Number of texts	Web scale	About 20	1
Types of natural language processing	Shallow processing (pattern matching or chunking)	Full processing (parsing, semantic interpretation)	Full processing
Attempt to express implicit information	No	Yes	Yes
Domain	Open	Focused	Open (high adaptation cost)
Knowledge integration	No	Yes	Yes (only for intersentences)
Style of texts	All genres	Expository writing	All genres (high adaptation cost)
Parts of the text that are interpreted	Partial	Partial	Whole

Table 2.1: Comparison of our task to the two major types of the machine reading tasks

shallow IE is that it scales and is domain-independent.

2.1.2 Comparison to Single-text NLU

The goals of single-text NLU and our task are similar in that both attempt to understand a whole text to build a knowledge representation of the text. However, they differ in terms of the representational language and the kinds of texts they read. The goal of our task is to produce graphical semantic representations and, therefore, is concerned with reading texts describing concepts and their relationships (e.g., technical documents). The NLU research, on the other hand, has tackled various genres (e.g., news articles and children books) while producing more expressive representations [1].

Despite a long history of research on NLU, it is still one of the unsolved challenges in AI because text is often too ambiguous to interpret accurately and omits information. Our approach to machine reading can alleviate this problem because it allows the system to rely on not a single, but multiple texts to extract the contents. Rather than trying to build systems that extract the *full* content of a single text, our approach is to build systems that extract a partial understanding from *many* texts (all on the same topic) and then integrate these into a single, coherent knowledge base. Our approach, therefore,

[1] For example, Epilog [128] uses Episodic Logic as its target representation to support reasoning with texts describing actions, situations, beliefs, and conditional statements. Episodic Logic roughly corresponds to the first-order logic with situational semantics. The UNO system [71] employs the expressive representational language, UNO, to handle adjectival and adverbial modification, set-related operations (e.g., disjunction and conjunction), negation, intervals, and uncertain or incomplete expressions.

partially shifts the burden of NLU from language processing tasks, which are known to be difficult, to knowledge integration, the task of combining fragments of information drawn from multiple texts.

In this dissertation, we present two methods to show how knowledge integration can improve text interpretation. Specifically, the first method fills in the gap of unspecified information in texts by using the information drawn from reading previous texts (Section 3.2.2.2). The second method improves the interpretation of one text by integrating it with other texts (Section 4.1).

2.2 Component Tasks in Our Machine Reading System

A machine reading system generally involves combining solutions to multiple component tasks. In this section, we first discuss the tasks that have been extensively studied in NLP: syntactic processing, semantic processing, and discourse-level processing. These tasks are important in machine reading as they produce the semantic representations of sentences along with the analysis of inter-sentential relationships (e.g., co-references). These tasks alone, however, are insufficient for building our machine reading system, which requires knowledge integration. In this dissertation, we use off-the-shelf components or simple, custom-built components for those tasks that are outside the scope of our research, focusing on developing novel methods for knowledge integration.

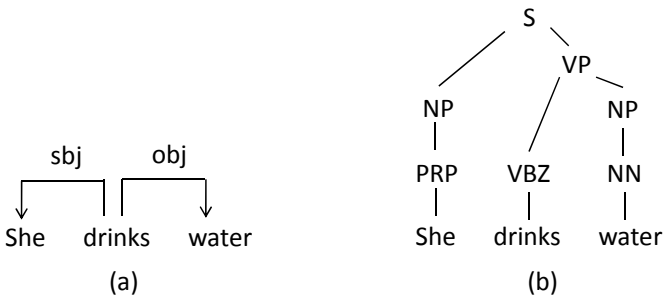


Figure 2.3: Two parse trees for *"She drinks water"*: (a) dependency parse (b) constituency parse

2.2.1 Syntactic Processing

The first task of a machine reading system is usually parsing a sentence into its syntactic representation. Two kinds of parsers have been widely used – dependency parsers and constituency parsers. Figure 2.3 shows an example of their outputs. They focus on different aspects of syntax: the constituency parse focuses on representing the phrase structures using non-terminals that correspond to each phrase in the sentence, whereas the dependency parse focuses on representing the dependency relationship among the words.

The dependency parse directly reflects the predicate-argument structure and, therefore, may be easier to convert to a logical form. Moreover, when a parse is fragmented, the fragments in the dependency parse may still be interpretable, which is not often the case in the constituency parse (see Figure 2.4). Some constituency parses, however, cannot be expressed by a dependency parse, and the accuracy of the state-of-art constituency parsers is slightly higher than the state-of-art dependency parsers [26]. Other syntactic

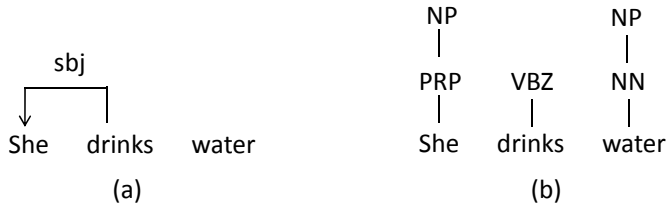


Figure 2.4: The fragmented parse trees

formalisms include LFG [21], GPSG [55], and HPSG [118]. In our project, we use a dependency parser, the Stanford Parser [79], because it is easy to use and has good performance.

Two major problems impair the performance of the state-of-the-art parsers which are mostly based on machine learning. First, the parsers are highly biased to the training domain; they perform well for the training domain, but the performance significantly drops when the domain is changed. For example, McClosky et al. [99] reports that the f-score [2] of the Charniak parser [29] trained on Wall Street Journal achieves 89.0% when the same corpus is used for the test, but the f-score significantly drops to 74.9% for a biomedical corpus, Genia [110]. Second, parsing requires semantic information, which is often unavailable during parsing. For example, consider the *with* prepositional phrase in the following sentences: (1) *"John bought the book with the money"* and (2) *"John bought the book with the ribbon"*. The two sentences look almost identical, but the attachment of the *with* prepositional

[2]F-score is the harmonic mean of precision and recall. Precision is a ratio of constituents in the parser's output that also appear in gold standard. Recall is a ratio of constituents in the gold standard parse that also appear in the parser's output.

phrase is different – the *with* phrase attaches to *bought* in (1) and *book* in (2). One way to address this problem is to include lexical semantics in parsing, but this requires a tremendous increase in the amount of training data.

For these reasons, it is often more advantageous to consider multiple parses rather than only the top-scored parse to improve the chance of acquiring the correct parse. Most state-of-the-art parsers can produce multiple candidate parses, but their representation, a list of full candidate parses, has several disadvantages. First, the number of the candidate parses is usually too many to be contained in the list. Considering most candidate parses are identical with only minor differences, the list representation is inefficient. Second, the list representation fails to capture dependencies in parsing. For example, in the following sentence – *"The chefs cook the fish."* – a dependency relationship may exist such that *chefs* is a syntactic subject of *cook* only when *cook* is a verb. The list form fails to capture this relationship.

To address these problems, we developed a method for succinctly representing multiple candidate parses along with the dependency relationship (Section 4.2.3.1).

2.2.2 Semantic Processing

Semantic processing is concerned with formally representing the meaning of a sentence. This task is especially important to enable the computer to understand sentences.

In this section, we discuss three major tasks in semantic processing:

word sense disambiguation, semantic role labeling, and temporal relation identification. Word sense disambiguation is the task of identifying the sense of a word and is primarily responsible for assigning a semantic concept to the nodes of a graphical representation. The other two tasks, semantic role labeling and temporal relation identification, are responsible for assinging semantic relations between two nodes.

Even though we introduce only three major tasks here, there exist other semantic tasks, including identifying event-event relations (e.g., causal/subevent relations) [14] and interpreting noun-noun compounds (e.g., identifying the relationship between the two nouns in, for example, *snow ball*) [84].

2.2.2.1 Word Sense Disambiguation

Word sense disambiguation (WSD) identifies the sense of a word in a given sentence when the word has multiple senses. For example, the two following sentences use the same word, *bank*, but in a different sense.

(1) John deposited money in the bank.
(2) John ran along the bank.

Bank in the first sentence means a financial institution, whereas *bank* in the second sentence means the sloping land beside a river. These senses are generally defined in a sense inventory such as WordNet [103] or OntoNotes [69] [3].

[3]The inventories could be an ontology of concepts. In this case, the WSD task is to assign a concept to the words (as in our system, described in Section 3.1).

One major approach to WSD is to use a supervised method in which the sense disambiguation model is learned from a corpus annotated with word senses. This approach, however, only performs well for the training domain [2], and an annotated corpus is expensive to construct. To address these problems, semi-supervised and unsupervised methods have been proposed [114] [148]. Semi-supervised methods require only a handful of seed annotations and extend them to produce more training data. Unsupervised methods, without relying on sense annotations, cluster words in terms of their sense. Semi-supervised and unsupervised methods, however, still require a large amount of text to derive significant statistics for both extending annotations and clustering.

A third approach is dictionary and knowledge-based methods. For example, the Lesk algorithm [86], the seminal dictionary-based method, chooses the senses of consecutive words whose dictionary definitions overlap most. This approach is domain-independent and does not require any additional text, but the performance may not be as accurate as the other approaches. In our project, we use SenseRelate [112], off-the-shelf WSD software that is based on the Lesk algorithm.

2.2.2.2 Semantic Role Labeling

Semantic role labeling (SRL) is the task of identifying an event and its participants (roles) in a given sentence. For example, consider the following sentence: *"John is eating spaghetti with his fork"*. In this sentence, the main

event is EATING and it has three roles, *agent*(John), *object*(spaghetti), and *instrument*(his fork).

Most SRL systems use one of three role sets: FrameNet [51], VerbNet [129], or PropBank [111]. FrameNet defines the frames that describe typical situations (e.g., APPEAL) and the lexical items that may trigger the frames (e.g., *appeal* and *plead*). It also defines the roles (called frame elements) for each frame. For example, the APPEAL frame has three core frame elements, *Convict*, *Decision*, and *Representative*, as in the following example sentence: *"John$_{[Convict]}$ will appeal$_{[APPEAL]}$ his conviction$_{[Decision]}$."* FrameNet also provides approximately 141,000 sentences annotated with the frames and their roles. One disadvantage of FrameNet's roles is that the role names are made specifically to the frames rather than being general across the frames, thereby failing to capture the similarity among the roles in different frames.

VerbNet, based on Levin's work [87], provides syntactic and semantic information about classes of English verbs. For example, the class, `leave-51.2` (which has two verb members, *abandon* and *split*), has a surface realization form, NP1 <verb> NP2 (e.g., *"we abandoned the area"*), in which NP1 and NP2 map to *theme* and *source*, respectively. Because of insufficient annotated text, VerbNet has been primarily used in unsupervised methods [97].

PropBank is a large-scale corpus that annotates semantic roles for all verbs in the Penn Treebank corpus (1M words). PropBank, inspired by VerbNet, defines predicates (e.g., `eat`) and their semantic roles (e.g., `Arg0` for *consumer/eater* and `Arg1` for *meal*). For example, the sentence, *"John ate*

the food", is annotated with `eat`(*ate*), `Arg0`(*John*), and `Arg1`(*the food*). The corpus has been a primary resource for SRL research because of its large scale and many other NLP programs trained on the Penn Treebank. PropBank has several advantages over FrameNet. First, the Penn Treebank corpus contains a greater variety of texts and genres than the manually selected sentences in FrameNet. Second, it uses a small number of roles as in VerbNet [4], which is advantageous for automated reasoning.

Most SRL approaches are based on supervised machine learning. The supervised method is exemplified by Gildea and Jurafsky [57], which identifies a predicate and its arguments and then labels the arguments using semantic roles. These steps have been followed by most subsequent methods. In CoNLL shared tasks in 2008 (which were based on PropBank), the F-score [5] of the best system was 81.75% for the in-domain evaluation, but significantly dropped to 69.06% for the out-of-domain evaluation [6]. In our project, we use a subset of semantic roles defined in the Component Library (summarized in Table 2.2) and the manually-built rules to assign them [7].

[4]The meaning of the role may differ for different predicates. For example, `arg2` means *destination* in the verb *bring*, whereas it means *extent* in the verb *rise*.

[5]The f-score is the harmonic mean of precision and recall. Precision is the percentage of semantic roles and predicates correctly assigned in the system's output. Recall is the percentage of semantic roles and predicates correctly assigned in the gold standard [136].

[6]The in-domain evaluation indicates that the training domain is the same as the test domain, and the out-of-domain evaluation indicates that the training domain is different.

[7]Most of the rules were developed as part of DARPA's Learning-by-Reading Project.

role	description
agent	*agent* initiates Event
base	Event references *base* as a major fixed thing
destination	Event ends at *destination*
donor	*donor* releases object of Event
instrument	*instrument* is used in Event
location	Event transpires at *location*
object	*object* is the main passive participant in Event
origin	Event begins at *origin*
path	Event transpires along *path* as path
raw-material	Event uses *raw-material* as input
recipient	*recipient* receives object of Event
result	*result* comes into being as a result of Event
site	*site* is a specific place of some effect of an Event

Table 2.2: The Component Library semantic roles used in our system

2.2.2.3 Temporal Relation Identification

A third task in semantic processing is to identify temporal relations among the events, which is important to organize event snippets coherently. This task has received much attention, particularly in question-answering and text summarization.

Prior to the emergence of large-scale annotated corpora, the main approach had been to use hand-built rules along with hand-crafted knowledge bases [3] [72] [63]. Like most manual approaches, however, they often failed to scale beyond the designated domains. More recently, several annotated corpora have been created, such as TimeBank [17], which is annotated with a markup language, TimeML.

TimeML provides two annotation tags. The first one is the EVENT

tag, which annotates a tensed verb, adverbs (e.g., *yesterday*, *this morning*), and nominals. The EVENT tag has various features such as tense, event class, grammatical aspect, polarity (i.e., positive or negative), any modal operators, and cardinality (whether the expression is mentioned more than once). The second tag is TLINK, which annotates the relation between an event and a temporal expression (e.g., in *"John studied this morning"*, *studied* temporally overlaps with *this morning*) and between two events (e.g., in *"Mary heard the scream "*, *heard* temporally overlaps with *the scream*).

Based on these annotated corpora, several supervised methods have been proposed [92] [28]. In the recent competition, TempEval 2010 [141], the top-performing system achieved the following accuracies [8] for identifying the temporal relation: 63% (between an event and a temporal expression), 80% (between an event and a document creation time [9]),and 56% (for two main events in consecutive sentences). Some semi-supervised [16] and unsupervised methods [82] have also been proposed. For example, Lapata and Lascarides [82] collect sentences containing discourse cues, such as *after* and *before* and then uses pairs of main clause and subordinate clause as the training examples. In our system, manually built simple rules are used to assign the temporal relations based on cue phrases such as *then*.

[8]The accuracy is defined as the ratio of the number of correct answers to the number of the answers made by the system.

[9]For example, in the sentence, *"President Barak Obama will visit South Korea next year"*, the time of *visit* is after the time when this document is created.

2.2.3 Discourse-level Processing

Discourse-level processing is concerned with analyzing inter-sentential relationships and, therefore, is useful for our knowledge integration task, which attempts to combine semantic representations of individual sentences. In this section, we review three major tasks at the discourse level: co-reference resolution, indirect anaphora resolution, and discourse relation identification.

2.2.3.1 Co-reference Resolution

Co-reference resolution is the task of identifying mentions that refer to the same object. The task is generally divided into several subtasks, such as definite noun phrase resolution, pronoun resolution, and event co-reference resolution, as illustrated in the following examples (the words in bold in (1) and (2) are co-referent with "the masked man", and the bold word in (3) is co-referent with "kidnapped").

> The masked man kidnapped the children.
> (1) **The kids** are found to be safe. (definite noun phrase resolution)
> (2) **They** are found to be safe. (pronoun resolution)
> (3) The children were **abducted** last night. (event co-reference resolution)

Early methods of coreference resolution were primarily based on linguistic properties. Hobbs' algorithm [66] resolves pronouns based on the hand-built heuristics that navigate through the parse tree. The centering theory [58]

models the relationship between the focus of attention (salient entities) in the discourse structure and referring expressions. This work has been the basis for several co-reference resolution methods, such as [20] and [139]. Tacitus [65] performs abductive reasoning, using domain-specific knowledge, to produce a coherent interpretation of a text, and co-references are resolved during this process.

Machine learning approaches have been emerging since the mid-90s; these methods have primarily been supervised methods. The seminal supervised method, [133], models co-reference resolution as a pairwise binary classification task – whether two mentions co-refer or not. This approach, however, may produce inconsistent outputs (e.g., X and Y co-refer, and Y and Z co-refer, but X and Z do not). To address this problem, clustering-based methods group co-referring mentions [23] [144] by considering global constraints. Some clustering methods are unsupervised, using a nonparametric Bayesian model [59] and the EM method [108].

For the ACE-2 dataset, the best supervised system achieves a MUC F-score [10] of 83.7%, while the unsupervised systems achieve a score of 62.3% – 64.2% [117]. Traditionally, entities (denoted by noun phrases or pronouns) have been the main focus in co-reference resolution research, but event co-reference resolution has been also gaining interest recently [13]. In our project, instead of state-of-the-art co-reference resolution software, we manually imple-

[10]MUC F-score is one of the commonly-used metrics for measuring a co-reference resolution system's performance. See [143] for the details of MUC F-score.

mented simple rules to resolve definite noun phrases.

A major bottleneck in co-reference resolution has been the lack of commonsense and encyclopedic knowledge. For example, consider the following text contained in the ACE-02 corpus (excerpted from [117]):

> **Israel** will ask the United States to delay a military strike against Iraq until **the Jewish state** is fully prepared for a possible Iraqi attack [....]. **Israel** is equipping **its** residents with gas masks and preparing kits with antidotes.

To correctly identify the mentions in bold as co-referring expressions, the system needs to know that Israel is the Jewish state, that Israel is a country, and that the country can have residents. It is, however, still an open problem to acquire knowledge of this type.

2.2.3.2 Indirect Anaphora Resolution

Indirect anaphora (also known as bridging reference or associative reference) is an expression that refers to a thing related to a previous mention. Indirect anaphora resolution is the task of identifying the antecedent and the relationship between the antecedent and the referring expression. For example, consider the sentence: *"John went to the office and opened the door."* In this sentence, *the door* does not refer to an arbitrary door, but the door of the office John went to – i.e., *the door* is a part of the antecedent, *the office*.

The core challenge to indirect anaphora resolution is how to supply

background information to infer unspecified relations. Early research attempted to address this challenge by manually building knowledge bases [135] [142], but these systems did not perform well. Some propose using the web as background knowledge [116] [22] [95]. For example, Poesio et al. [116] use the query, *the X of the Y* (X: antecedent, Y: referring expression) to detect the mereological (part-whole) relation (e.g., *"the door of the office"*) and Markert and Nissim [95] use the query, *X and the other Y*, to identify the taxonomic relation (e.g., *the sedan and the other cars*).

WordNet is also often used to provide semantic relations between the words, such as hypernym, meronym, and entailment [116] [47]. Viera and Poesio [142], however, report that WordNet misses many pieces of useful information; 62% of meronomy relations in Penn Treebank I that is annotated with semantic relations do not appear in WordNet. As in other NLP tasks, the problem of acquiring background knowledge remains unsolved. In our project, we supply the background knowledge to indirect anaphora resolution by using the knowledge acquired from reading previous texts.

2.2.3.3 Discourse Relation Identification

Discourse relations describe the relationship among the different segments in discourse. For example, in the following sentences, *"Be quiet! I am writing a dissertation"*, the second sentence explains the reason for uttering the first sentence. Several theories have provided a small set of discourse relations [93] [64] [83] [80]. By grouping these relations, Marcu and Echihabi

Discourse relation	Explanation	Cue words
Contrast	S1 contrasts to S2	*but*, *although*, *even though*
Cause-Explanation-Evidence	S1 causes/explains S2	*because*, *therefore*, *thus*
Elaboration	S1 is elaborated by S2	*for example*
Condition	if S1, S2	*if ... then*

Table 2.3: Four discourse relations. S1 and S2 are the two segments.

[94] present four high-level discourse relations: Contrast, Cause-Explanation-Evidence, Elaboration, and Condition. Table 2.3 explains these relations and their cue words.

Two major corpora annotated with discourse relations are Penn Discourse Tree Bank(PDTB) [119] and RST (Rhetorical Structure Theory) Discourse Treebank (RST-DT) [25]. Both corpora are based on Wall Street Journal but are annotated with different discourse relations: PDTB defines its own relation set while RST-DT is annotated with the RST relations [93]. Based on these corpora, several machine learning algorithms have been proposed, such as supervised methods [40] [115] [89], semi-supervised methods [62], and unsupervised methods [94]. Pitler et al. [115] reports that their state-of-the-art supervised method, when trained and tested on PDTB, achieves 74.74% accuracy (93.09% accuracy for explicitly stated discourse markers). In our project, we use simple rules to map the cue words into the discourse relations.

2.3 Knowledge Integration in Machine Reading

In Section 2.2, we reviewed three major tasks in NLP – syntactic processing, semantic processing, and discourse-level processing, all of which are important in our machine reading task because they produce the semantic representations of individual sentences (from syntactic and semantic processing) along with their inter-sentential relationships (from discourse-level processing). These tasks alone are, however, insufficient to build a useful, coherent knowledge base (such as the one shown in Figure 2.2). The individual semantic representations still need to be combined into a coherent knowledge base, along with background knowledge the system may possess. This task, knowledge integration, is non-trivial, requiring sophisticated methods beyond simply aggregating the individual semantic representations.

In this section, we introduce our knowledge integration task in detail (Section 2.3.1) and then explain why knowledge integration is critical to machine reading (Section 2.3.2). We further present the challenges in knowledge integration (Section 2.3.3). Knowledge integration has been studied in other fields of AI; we review these tasks and discuss how they differ from our task (Section 2.3.4). Finally, we introduce the two goals that this dissertation addresses: coherent combination of knowledge snippets and application of knowledge integration to text interpretation (Section 2.3.5).

2.3.1 Task Definition

Knowledge integration combines knowledge snippets (semantic representations of sentences or phrases), along with background knowledge the system may possess, into a single coherent knowledge base. Our approach to Machine reading requires performing three knowledge integration tasks: combining sentences within the same text, combining sentences across multiple texts, and combining texts with the background knowledge base. Each task requires different processing.

Knowledge integration within the same text is needed when consecutive sentences in the text are combined. Identifying co-references and discourse relations may benefit this task but several problems remain. First, to help readers' understanding, the same concept may be explained several different ways throughout the text. For example, some sentences in the texts – usually the first sentences – may describe a concept at the high level, while subsequent sentences may elaborate the concept further. Aligning the semantic representations of these two text segments is challenging; even though they may be formulated differently, they express the same concept. Section 2.3.3 explains this challenge in further detail.

Another problem in combining consecutive sentences is that texts omit a vast amount of background information that human readers can easily infer. This omitted information needs to be explicitly represented to coherently combine the semantic representations of the sentences.

Knowledge integration across multiple texts also requires combining similar content that is represented differently. This task may be more difficult than within-document knowledge integration because of a higher variance among the texts. Moreover, as the corpus expands, knowledge integration may need to distinguish subject matter more precisely. For example, knowledge extracted from a text about the human heart should be distinguished from, for example, the amphibian heart [11]. Efficiency is another issue, especially when the size of the corpus is large. Finally, conflict among the knowledge snippets should be appropriately resolved [12].

Knowledge integration between texts and knowledge bases occurs in two places: when the system uses the knowledge base to interpret the texts during reading and when the system updates its own knowledge base with new information extracted from the texts. During reading, the system may use the knowledge base, for example, to correct interpretations produced by NLP or to reveal unspecified information in the texts. One challenge to this task is to identify the knowledge relevant to the texts from the knowledge base. This task could be difficult, especially when the size of the knowledge base is large. The knowledge base update requires performing several tasks: identifying the parts that should be updated, detecting possible side effects (e.g., inconsistency occurrences), and resolving the side effects properly [106].

[11] For example, a text about the human heart would state that the heart has four chambers, while the text about the amphibian heart would state that the heart has three chambers. The representations of these two pieces of information should be separately stored in the knowledge base to avoid inconsistency in the knowledge base.

[12] This conflict may occur because of the mistakes by the authors of the texts.

Finally, all these knowledge integration tasks require addressing one common issue: the semantic representations produced by NLP (which is given to knowledge integration) could be poorly formulated because of the NLP errors. It is important for knowledge integration to filter out erroneous representations to maintain a high quality resulting knowledge base.

In this dissertation, we present a proof-of-concept system, Kleo, which performs some of the knowledge integration tasks described above. In particular, Kleo attempts to resolve granularity mismatches among the semantic representations that express the same content but with different levels of detail. This method is used in both Kleo's within- and across-document knowledge integration. Kleo also infers unspecified information in texts by using content acquired from reading previous texts. We also studied the issue of managing uncertainty caused by NLP errors. Issues in knowledge integration that are not addressed in this dissertation are presented in the future work section (Chapter 5).

2.3.2 Importance of Knowledge Integration

Knowledge integration is important in machine reading for three reasons. First, knowledge integration could improve the reasoning capability of the output knowledge base. Second, knowledge integration could improve language interpretation capability by accessing a variety of information sources (e.g., other texts in the corpus and external knowledge resources). Third, knowledge integration could facilitate knowledge base management by storing

the redundant information extracted from texts only once.

2.3.2.1 Knowledge integration improves reasoning

Machine reading systems often produce a fragmented knowledge base, in which the knowledge snippets are largely unconnected [9]. Two issues are responsible for this fragmented knowledge base: failure of relating the knowledge snippets and NLP errors. [13]

A fragmented knowledge base is not suited for reasoning tasks such as question-answering. For instance, consider a simple example of a fragmented knowledge base shown in (a) in Figure 2.5. This knowledge base would fail to answer the following question, *"what is the relationship between the piston and the engine?"*, because no semantic relation exists between `engine-1` and `piston-2`. To answer this question, the two cylinder instances should be aligned as in (b) to represent that the piston is contained in the engine.

In Chapter 3, we present several knowledge integration methods for reducing knowledge fragmentation.

2.3.2.2 Knowledge integration improves language interpretation

Beyond improving reasoning capability, knowledge integration can also improve language interpretation; knowledge integration brings together multiple pieces of knowledge from various sources (such as texts and external

[13] For example, parsing errors may cause fragmented graphical representations, in which many single nodes are unconnected.

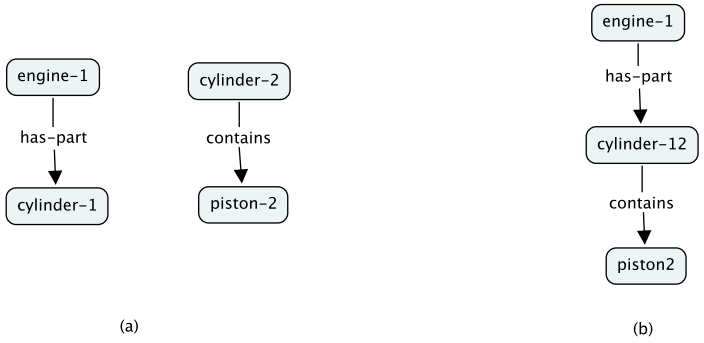


Figure 2.5: (a) a fragmented knowledge base and (b) an integrated knowledge base

knowledge bases) that may be useful for interpreting the current text.

For example, consider interpreting the following sentence: *"Combustion of gasoline produces energy"*. To interpret *"of"*, the system should know the relation between the gasoline and the combustion process, because *"of"* can represent different semantic relations depending on its context (e.g., part-whole relation as in *"the door of the building"*, taxonomic relation as in *"the process of combustion"*, or locative relation as in *"the combustion of the engine"*). This knowledge can be provided by two sources. The first source is another sentence that states, for example, *"The engines combust gasoline"*; this sentence explicitly states that the gasoline is the *object* of the combustion process. The second source is a background knowledge base that may contain knowledge that the gasoline can be the object of the combustion event.

One challenge to using knowledge integration for text interpretation is the delay of ambiguity resolution during the NLP tasks so that knowledge integration can perform ambiguity resolution more reliably by exploiting

more information sources. In Chapter 4, we present a method for efficiently postponing ambiguity resolution and explore several information sources that knowledge integration could exploit for ambiguity resolution.

2.3.2.3 Knowledge integration facilitates knowledge base management

Reading through a corpus, the system should update its own knowledge base with new information extracted from the texts. This update process, however, may cause inconsistency in the knowledge base when multiple instances of the same information are maintained redundantly. For example, imagine a case in which the old information should be replaced with new information. If the knowledge base contains multiple instances of the old information, all instances should be searched and updated. If only some of the instances are updated, the knowledge base will be inconsistent. Knowledge integration can help this update process by allowing the system to maintain only a single instance of redundant representations.

2.3.3 Challenges in Knowledge Integration

Two knowledge integration operations are useful for aligning knowledge snippets. The first operation is aligning parts that represent the same meaning. For example, given the two knowledge snippets shown in (a) in Figure 2.6, this operation would combine `gasoline-1` with `gasoline-2` (because they represent the same concept) to produce the representation shown in (b). The second operation is inferring new information to establish more connec-

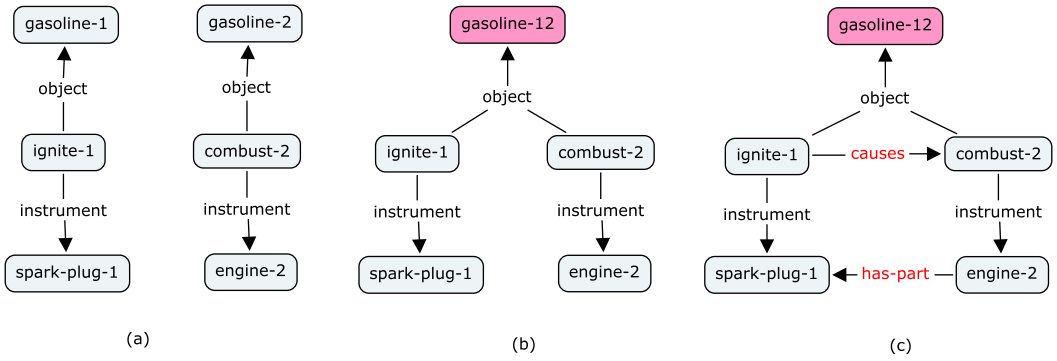


Figure 2.6: Two knowledge integration operations

tions among the knowledge snippets. (c) in Figure 2.6 shows a representation resulting from applying this operation to (b); the system further infers that `ignite-1` causes `combust-2` and that `spark-plug-1` is a part of `engine-2`. However, these two operations are challenging.

2.3.3.1 Aligning representations of the same content

Aligning representations that express the same meaning is challenging, because the content can be expressed in a variety of forms. For example, consider the sentence: *(1) "Blood moves from the heart to the body."* The movement event in this sentence can be expressed in a variety of ways, including:

(2) "Blood flows/travels from the heart to the body."
(3) "The heart pumps blood to the body."
(4) "Blood moves from the heart to the lung and then from the lung to the body."

(5) "Blood circulates throughout the body."

(6) "The body receives blood from the heart."

Because of the variety of surface forms, naïve alignment often fails. For example, consider combining (1) and (4), whose semantic representations are shown in (a) in Figure 2.7. Simple graph matching shown in (b) – which is commonly used in knowledge integration – combines the two input representations by aligning their maximal common subgraph. This operation, however, does not consider the granularity mismatch of the input representations, thereby producing a representation that contains a movement with two destinations [14]. Rather, the two input representations should be combined flexibly as in (c), which identifies the granularity mismatch and, based on the identification, combines the three movements using the *subevent* relations.

The following are common cases that require flexible matching:

1. **Synonyms:** Different words can be used. For example, (2) uses *flows/travels* instead of *move*.

2. **Lexical compression:** Some lexemes represent a complex idea; for example, *pumps* in (3), when used with the preposition *to*, implicitly expresses "the pumping causes <the object of the pump> to move". This movement, which is important information to be aligned with (1), is unspecified in (3).

[14]The movement event should have only one destination.

3. **Granularity mismatch:** The same idea can be presented in different levels of detail. For example, the description of blood circulation in (4) is more fine-grained than (1), and (5) is more coarse-grained than (1).

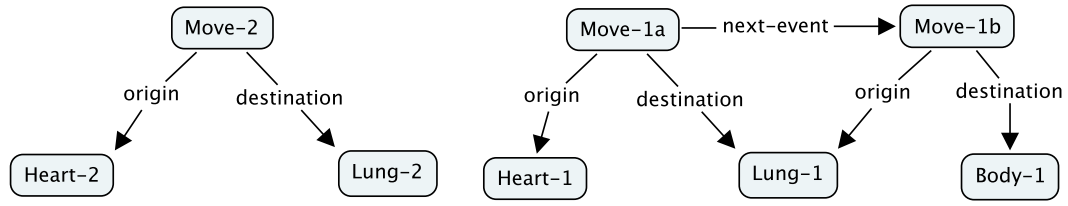
4. **Viewpoint difference:** The same content could be described from different perspectives. For example, (6) describes the blood circulation as a transfer of possession.

In this dissertation, we particularly focus on the granularity mismatch case, such as the one shown in Figure 2.7.

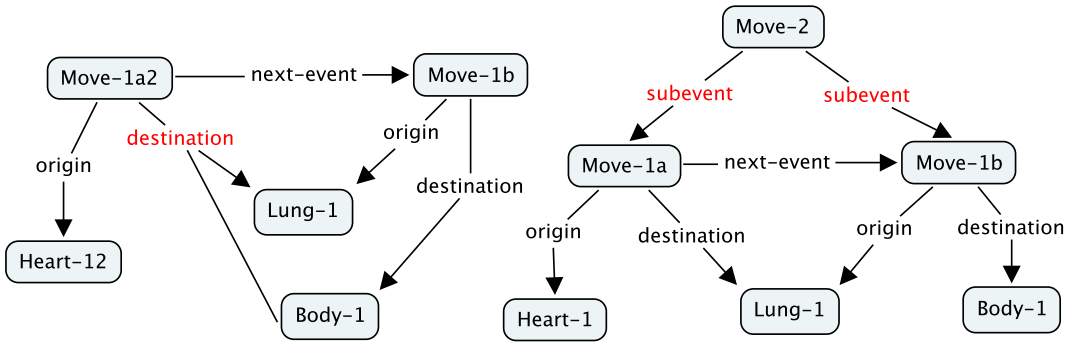
2.3.3.2 Inferring unspecified information

Inferring unspecified information in texts has been one of the core problems in text understanding, because a great deal of information is often unspecified in texts. Despite several approaches to provide such background information (e.g., manual construction [85] [51] [103] [129] [11], automated mining [127] [32] [88], or construction by volunteers [130] [31]), there has yet to be a knowledge base created that provides enough background information for text understanding. Moreover, even though large-scale knowledge bases are available, it is difficult to determine what knowledge is relevant to understanding the current text.

In Chapter 3, we present an approach to this challenge based on multi-text reading. Because our system reads many texts on the same topic, the information previously acquired from other texts can help in reading the cur-



(a) two input knowledge representations



(b) simple graph matching

(c) flexible semantic matching

Figure 2.7: **(a)** Semantic representations for (1) and (4), which represent the blood circulation with different granularity; **(b)** Simple graph matching: `Move-1a2` has two destinations (incorrect); **(c)** Flexible matching: the subevent relations are established from `Move-2` to `Move-1a` and `Move-1b` (correct)

rent one. For example, consider the sentence: *"The piston compresses the gasoline inside the engine"*. The interpretation of this sentence requires resolving the indirect anaphora, *the engine*. The knowledge necessary for this resolution can be provided by reading another phrase. For example, *"The piston in the engine"* states the piston is the *part* of the engine.

2.3.4 Knowledge Integration in Other Fields of AI

Knowledge integration – combining different pieces of knowledge – is a central task in human reasoning. For example, Conceptual Blending [50], a seminal computational model for human reasoning, claims that humans constantly combine/blend pieces of knowledge (that are previously unrelated) to create new concepts and knowledge. For this reason, several fields of AI have studied knowledge integration extensively, and it is important to know how our knowledge integration task is different from knowledge integration in the other fields of AI.

2.3.4.1 Research in knowledge-based systems

We review three areas in knowledge-based systems research that are concerned with knowledge integration: ontology alignment/merging, knowledge acquisition, and knowledge-based question-answering.

Ontology alignment and merging

Ontology alignment/merging (OAM) [45] is concerned with aligning different ontologies or database schemas. Similar to our task that combines

semantic representations flexibly beyond the surface forms, OAM may combine ontologies (formulated by the different parties) that conceptualize the same world differently. Our task and OAM differ in the following ways. First, OAM generally combines knowledge structures that are manually constructed; therefore, OAM is not concerned with the impoverished representations, which are often the result of NLP errors. Second, OAM systems generally only deal with ontologies about entities and their relations (e.g., GLUE [38]). In contrast, our task attempts to combine various types of knowledge, such as events and their causal/temporal relations, which are often expressed in natural texts.

Knowledge acquisition

Knowledge acquisition is concerned with building a knowledge base with information elicited from domain experts. One important step in knowledge acquisition is to assimilate new elicited information into the knowledge base [106]. This assimilation task is related to our task in that both attempt to update the knowledge base with new information.

They are different in the following ways. First, our task, which deals with information extracted from texts, must address NLP-related issues such as handling wrong representations or finding informative texts, whereas knowledge acquisition focuses on developing elicitation methods from domain experts. Second, the types of knowledge the two tasks deal with are different. Our task deals with knowledge about concepts and their relations, whereas knowledge acquisition may deal with more varied types of knowledge, such as procedural knowledge.

Knowledge-based question-answering

Knowledge-based question-answering attempts to solve questions by reasoning with the knowledge base [54]. Some systems even accept questions in the form of natural language. This task is similar to our task in that it performs NLP to interpret the questions and should combine the formal representation of the problem description with the knowledge base. However, our task differs from question-answering task in that our task is concerned not only with combining texts with the knowledge base, but also with combining sentences to one another. For this reason, a focus in our task is aligning redundant information across multiple texts, which is not done in question-answering.

2.3.4.2 Research in discourse-level processing

As illustrated in Section 2.2.3, discourse-level processing in NLP could benefit knowledge integration by providing inter-sentential information, such as co-references and discourse relations. The goal of these tasks is, however, to analyze inter-sentential relationships, not necessarily focusing on producing actual combined semantic representations. As such, these NLP tasks are different from our task in several ways.

First, knowledge integration requires aligning the semantic representations beyond co-references. Most co-reference resolution methods focus on identifying the mentions that refer to the same thing (mostly an entity) and can therefore inform knowledge integration of which (entity) nodes in our

graphical representation should be aligned. Co-reference resolution, however, does not address aligning whole meaning representations, as Section 2.3.3.1 describes.

Second, some NLP analysis is non-trivial to transform into knowledge integration operations. For example, consider the discourse relation, *Elaborate*, between the two following sentences:

(1) Hearts generally have several muscular chambers.
(2) For example, the human heart has four chambers.

Elaborate indicates that the second sentence provides further detail for the first sentence. This relation, however, does not tell how their semantic representations should be combined – e.g., inferring *isa* relations from *the heart* in (1) to *the human heart* in (2) and from *chambers* in (1) to each of *four chambers* in (2).

Finally, our knowledge integration attempts to explicitly represent unspecified relationships among knowledge snippets because texts omit a vast amount of trivial information. For example, given the sentence, *"Hearts are pumps that propel blood around the body"*, the following facts should be further inferred: *hearts* perform pumping, the pumping causes *propel*, and *blood* is originally in *heart*. To perform this task, the system needs a general background knowledge base to provide rich background information. Existing NLP methods (such as indirect anaphora resolution, causal/temporal relation discovery and discourse-relation identification) can help identify parts of those

unspecified relations, but recent methods focus only on a small subset of simple relations. For example, most indirect anaphora resolution methods focus only on identifying hypernym and meronym.

2.3.5 Our Project Description

This dissertation has two goals concerning knowledge integration in machine reading.

The first goal is to show that sophisticated knowledge integration enhances the reasoning power of an output knowledge base. We particularly focus on three challenges in knowledge integration: resolving granularity mismatches between knowledge snippets, combining sentences within a text, and combining sentences across texts. Then, we evaluate our methods by measuring the quality of the knowledge base produced by our approach.

The second goal is to show that knowledge integration could improve language interpretation. Because knowledge integration can access a variety of information sources (e.g., other texts and external knowledge resources), the additional information may benefit language interpretation. A key challenge to this approach is to develop a method for delaying ambiguity resolution in NLP components so that knowledge integration can perform the ambiguity resolution with high accuracy. We also explore various information sources that knowledge integration could exploit for its ambiguity resolution. We particularly focus on redundancy across multiple texts and external knowledge resources.

2.3.6 Summary

In this chapter, we defined our machine reading task by comparing it against other types of machine reading. Then, we discussed component tasks in a machine reading system that have been extensively studied in NLP. These tasks alone, however, are insufficient to build a useful knowledge base. Knowledge integration is also needed to build a single coherent knowledge base from the knowledge snippets produced by NLP components. Finally, we introduced the knowledge integration task.

Chapter 3

Knowledge Integration: Combining Knowledge Snippets Coherently

Our knowledge integration research was motivated by a major drawback of our early machine reading system, Mobius [1]. The knowledge base built by Mobius was highly fragmented, containing many unrelated knowledge snippets. As explained in Chapter 2 (see Figure 2.5), a fragmented knowledge base fails to deliver a coherent set of knowledge representations, making it unsuitable for use by a reasoning system. To address knowledge fragmentation, we developed a proof-of-concept machine reading system, Kleo, which has sophisticated knowledge integration facilities [73].

Kleo performs two types of knowledge integration. The first type is sentence-to-sentence knowledge integration, which combines information extracted from individual sentences within one text to form a representation of the whole text. In this knowledge integration, Kleo uses knowledge learned from previous reading to fill the gaps of the unspecified information in the text. The second type is text-to-text knowledge integration, which combines

[1] Mobius was built by several research institutes during the DARPA Learning-by-Reading project [9]. The participants were UT Austin, ISI, BBN, SRI, and Noah Friedland.

the output of sentence-to-sentence knowledge integration (i.e., the representations for the individual texts) into a single coherent knowledge base. The core component of these two knowledge integration facilities is a graph matcher, which can align two graphical representations despite their mismatches in levels of granularity. Figure 2.7 (c) shows an example of our matcher's operation. Our evaluation shows that this approach to knowledge integration is both feasible and promising.

This chapter is organized as follows. We introduce Kleo in Section 3.1 and explain its knowledge integration facility in Section 3.2. In Section 3.3, we present our evaluation results, which shows our knowledge integration methods are effective for resolving knowledge fragmentation. Finally, we present related work in Section 3.4.

3.1 Kleo

To provide the background of our approach on knowledge integration, this section presents the general overview of a machine reading system, Kleo, in which our approach is implemented.

Figure 3.1 shows the architecture of Kleo. Kleo consists of two main components: the NL component and the KI component. Given a corpus of texts, the NL component produces the semantic representation of the individual sentences, and the KI component combines these representations into a single coherent knowledge base. Note the cycle in which the NL component accesses the knowledge base and the KI component updates the knowledge base.

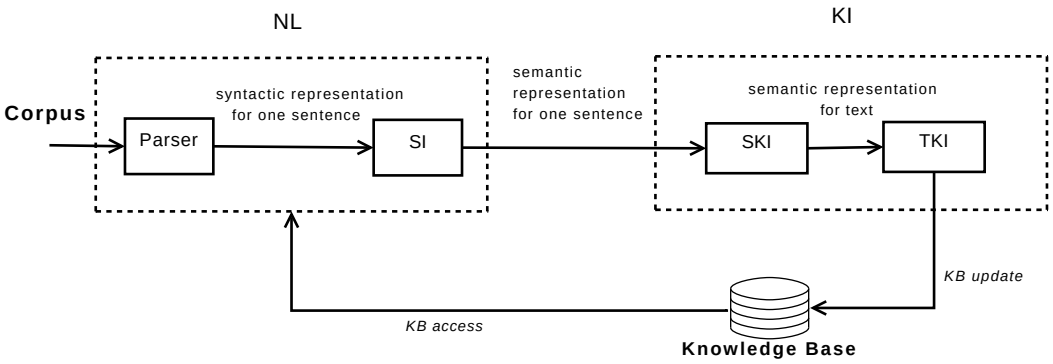


Figure 3.1: Architecture of Kleo: The arrows represent data flow.

It constitutes a bootstrap learning cycle where reading extends the knowledge base (KB update) and the extended knowledge base is in turn used in language interpretation (KB access). Currently, Kleo uses the knowledge base for only small task of the NL component – indirect anaphora resolution.

The next subsections explain the components of Kleo in detail, using this example text:

S1: The engine's piston compresses the gasoline.

S2: Then, combustion of gasoline occurs inside the engine's cylinder.

3.1.1 The NL Component

The NL component performs two main processes: parsing and semantic interpretation.

```
det(engine-2, The-1)
poss(piston-4, engine-2)
nsubj(compresses-5, piston-4)
det(gasoline-7, the-6)
dobj(compresses-5, gasoline-7)
```

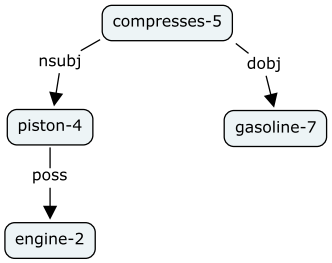


Figure 3.2: The dependency parse for S1 produced by the Stanford Parser

3.1.1.1 Parser

Kleo uses the off-the-shelf dependency parser, the Stanford Parser [79]. The Stanford Parser is fast and provides broad coverage and high accuracy. Marneffe et al. [96] provide the inventory of the 48 dependency relations used in the Stanford Parser. Figure 3.2 shows the output from the Stanford Parser for S1 [2].

3.1.1.2 Semantic Interpreter

The semantic interpreter converts the dependency parse into the corresponding semantic representation. Figure 3.3 shows the semantic representations for S1 and S2. As the figure shows, two types of semantic information are produced: Each node is assigned a type (e.g. DEVICE for `piston-4`), and semantic relations relate two nodes (e.g. `piston-4` is the *instrument* of `compresses-5`). This semantic information – the types and the semantic relations – are defined in a formal ontology, the Component Library [11]. The

[2]The number in a variable represents the location of the variable in the sentence. For example, occur-5 represents that the variable is from the 5th word from the first word.

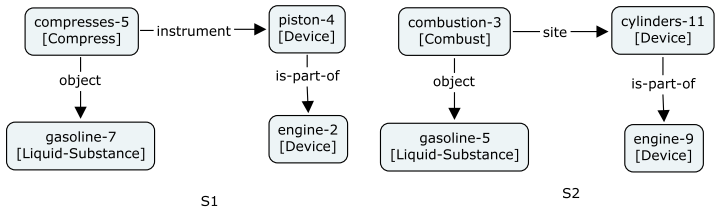


Figure 3.3: The semantic representations for S1 and S2

Component Library provides approximately 800 domain-independent concepts and 80 semantic relations.

The semantic interpreter operates in the bottom-up fashion: it resolves the leaf nodes first and then proceeds to the root node. For each node, Kleo assigns the most appropriate type (using off-the-shelf word sense disambiguation software [112] and wordnet-to-ComponentLibrary mappings) and converts the syntactic dependency relations into a semantic relation defined in the Component Library using the hand-written rules [3]. For example, for `compresses-5`, Kleo chooses a type, COMPRESS (see (c) in Figure 3.4), and resolves the syntactic relations such as (compresses-5 nsubj piston-4) using a rule

```
Produce (X instrument Y) for (X nsubj Y)
if the type of X is Compress and Y is inanimate.
```

Figure 3.4 shows each step in which our semantic interpreter converts the dependency tree in Figure 3.2 into the semantic representation. Kleo is not limited to these manually-written rules. More sophisticated methods (e.g., machine learning based approach) could be used for relation assignment.

[3]Most of the rules were formulated for the development of Mobius.

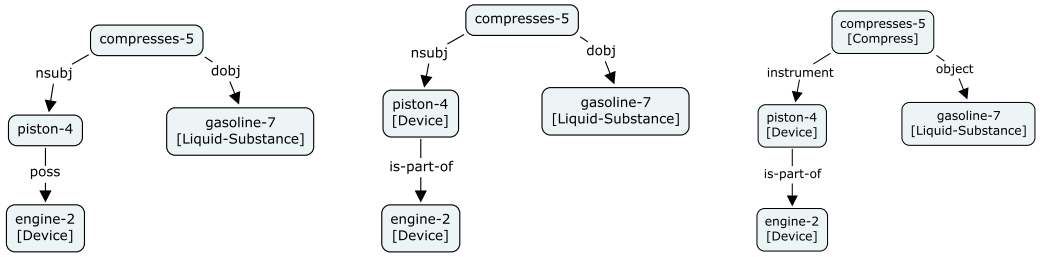


Figure 3.4: The intermediate steps of converting the dependency parse for S1 to the semantic representation. (a) `engine-2` and `gasoline-7` are assigned a type (b) `piston-4` is assigned a type and its dependency relation, *poss* is converted into *is-part-of* (c) `compresses-5` is assigned type and its dependency relations, *nsubj* and *dobj*, are resolved

After resolving all nodes and the dependency relations, Kleo elaborates the semantic representation further using the background knowledge base, which may resolve some indirect anaphora. For example, `gasoline-7` may be connected to `engine-2` through *encloses* by background knowledge, (Engine encloses Gasoline), acquired possibly from reading *"gasoline in the engine"* previously.

Table 3.1 shows the performance of the NL component in the parsing, the type assignment, and the semantic relation assignment. For parsing, we evaluated the four main processes : whether the parser correctly identifies two phrases conjunctively conjoined (**conjunctives**), whether it correctly attaches the prepositional phrases (**pp-attachment**), whether it correctly interprets "to *verb*" as either purposive clause modifier or to infinitive (**to verb**) and whether it correctly interprets "be *verb*" as one of the passive construction, an existential verb, a copula or the progressive construction (**be verb**). Its

accuracy is shown in the percentage. For the type assignment, we evaluated three cases: whether the semantic interpreter correctly identifies the types from the Component Library, for the nouns except the the nominalized forms (**noun**), the verbs (**verb**), and the nominalized verbs (**nominal forms**). For each word, Kleo's semantic interpreter returns a list of the candidate types with the score. The percentage indicates the number of times in which the candidate list contains the correct type. The number in the parenthesis is the average ranking of the correct type in the list. Finally, for semantic relation assignment, we evaluated three cases: whether the semantic interpreter correctly resolves the case roles for a verb (except the nominal cases) (**case roles**) and for a nominalized verb (**case roles for nominal verb**) and whether it correctly interprets the prepositional phrases (**PP**). Their accuracy is shown as a percentage.

Parsing	
conjunctives	56.2%
pp-attachment	89.5%
to verb	67.7%
be verb	94.0%

Type assignment	
noun	89.4% (1.2)
verb	61.0% (1.3)
nominal form	35.0% (1.1)

Semantic relation assignment	
case roles	67%
case roles for nominal verb	20%
PP	18.6%

Table 3.1: The performance of the NL component in Kleo measured for 25 texts in each domain of the heart and the engine.

3.1.2 The KI Component

Kleo performs two types of knowledge integration: sentence-to-sentence knowledge integration (SKI) and text-to-text knowledge integration (TKI). SKI combines the semantic representations of the sentences within the same text into a coherent whole. The output of SKI, therefore, represents the meaning of the whole text. TKI combines these outputs of SKI across texts. Section 3.4 presents these knowledge integration facilities in detail.

3.1.2.1 Sentence-to-Sentence Knowledge Integration

The goal of SKI is, given a text, to integrate semantic representations for individual sentences into a representation for the whole text in which entities and events mentioned in the text are coherently organized. For example, SKI combines S1 and S2 in Figure 3.3 by identifying

- `engine-8` is joined [134] with `engine-2` and `gasoline-4` is joined with `gasoline-6` because they derive from the same word class. [4]
- `cylinders-10` typically encloses `piston-3` (using background knowledge)
- `gasoline-5` is typically compressed in `cylinders-11`. (using background knowledge)

[4]This is a weak heuristic for resolving co-references. In our research, we are not focused on co-reference resolutions. Kleo could incorporate other state-of-the-art methods [105] for co-reference resolution, and is not limited to this weak heuristic.

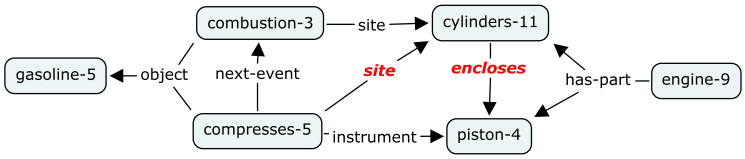


Figure 3.5: The outcome representation from combining S1 and S2 in Figure 3.3

- *"then"* implies that `combustion-1` occurs after `compresses-4`.

and produces a representation shown in Figure 3.5.

3.1.2.2 Text-to-Text Knowledge Integration

TKI integrates new information acquired from SKI with the knowledge base of Kleo. The goal of this step is to relate knowledge across texts. Because texts often contain redundant information, it is important to align them to produce a coherent knowledge base. Another important issue is the scalability of the run-time performance of TKI, because the size of the knowledge base could be large.

3.1.3 Knowledge Base

The knowledge base is extended with the output of TKI – the representation for a current text. At the same time, the knowledge base is used to help read a text. This constitutes a bootstrap cycle where reading extends the knowledge base and then the extended knowledge base enhances the subsequent reading.

The knowledge base initially contains only the Component Library,

which provides rich representations of general events and entities, such as ENTER and CONTAINER. For example, if Kleo reads *"fuel enters the cylinder"*, it draws on knowledge of ENTER to infer that a cylinder is a container (the base of all ENTER events) and that the fuel passes through a portal of the cylinder enroute from the outside of the cylinder to the inside.

3.2 Our Approach

This section presents our approach on knowledge integration. We first present a graph matcher in Section 3.2.1 which is a core component of SKI and TKI. Then, Section 3.4.2 and 3.4.3 present algorithms for SKI and TKI which are designed on top of the graph matcher.

3.2.1 Graph Matcher

The central issue in both SKI and TKI is combining multiple representations of knowledge into a coherent whole. The basic operation is graph join [134], but complications commonly arise because the representations fail to align perfectly. Our graph matcher attempts to resolve those mismatches.

Based on the earlier work on semantic matching [151], our graph matcher handles a common source of mismatch, shifts in granularity, such as the example shown in (a) in Figure 2.7. It is important in both SKI and TKI to resolve granularity mismatches because representations can differ in their level of detail between two sentences and between a text and a knowledge base.

To identify common types of granularity mismatches, we performed the

Filtering	Some information is omitted · The blood flows to the lung *to pick up oxygen* and then circulates around the body · The blood flows to the lung and then circulates around the body.
Generalization	Several similar pieces of information are generalized · An engine has *cylinders and pistons*. · An engine consists of several *parts*.
Abstraction	Two things that are spatially or temporally consecutive are viewed as one thing. · Blood *moves from the heart to the lung*, and *then from the lung to the body*. · Blood moves from the heart to the body.
Co-reference across granularity difference	Co-referenced entities differ in the level of granularity · *The cylinder in the engine* takes in gasoline. · *The engine* takes in gasoline.

Table 3.2: Types of granularity differences with examples: Granularity differences are shown in *italics*. The top sentence in the examples is fine-grained, the bottom coarse-grained.

following analysis. We selected approximately 50 general texts from Wikipedia, and manually performed knowledge integration on the sentences of the texts to identify common cases of granularity mismatches. Within paragraphs, a major source of mismatches was between the topic sentences (generally the beginning sentences) and the body of the paragraph. The body is often a fine-grained representation of the topic sentence. This analysis revealed four major types of the granularity mismatches: filtering, generalization, abstraction, co-reference across granularity difference (Table 3.2) [75]. For each type, we developed a graph representation pattern that characterizes it [76]. We describe the patterns along with examples shortly.

Algorithm. 1 outlines the operation of our matcher. The algorithm first selects initial mappings (seed nodes) using heuristics and then extends the mappings from the seed nodes.

Algorithm 1 Graph matching

(1) The matcher identifies the initial mappings between the two graphs using two heuristics.
(2) The matcher finds more mappings from the previous ones using the patterns until no more mappings are found.

Identifying seed nodes

The matcher uses two heuristics to identify seed nodes. In the description of the heuristics, let *node1* be a node from the first input graph, and *node2* from the second.

Heuristic 1. Node1 is mapped with node2 if their types are taxonomically related entities and they originate from the same word class.

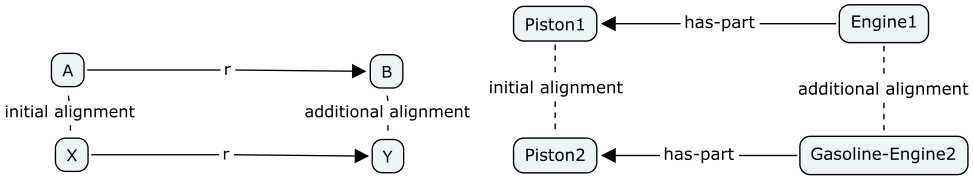
Heuristic 2. Node1 is mapped with node2 if their types are events, node1 subsumes node2 (or *vice versa*) and the case roles of node1 subsume the roles of node2 (or *vice versa*).

As an example of heuristic2, consider two representations, *"Fuel moves into an engine"* (Move-Into1 object Fuel1) (Move-Into1 destination Engine1) and *"Engine takes in gasoline"* (Take-In2 object Gasoline2) (Take-In2 destination Engine1). Heuristic2 chooses (Move-Into1, Take-In2) as a pair of seed nodes because, in the Component Library, TAKE-IN and GASOLINE are subclasses of MOVE-IN and FUEL, respectively. It is important to check the case roles because the case roles restrict the meaning of the events. To identify more seed nodes, state-of-the-art methods for co-reference resolution such as [105] could be incorporated.

Extending mappings

The mappings are recursively extended from the seed nodes by *extension pattern* rules. In the description of the patterns, G1 and G2 refer to the two input graphs, and A and X refer to nodes in G1 and G2 that are already mapped to each other. The graphic representations on the patterns are shown on the left side and their examples on the right side.

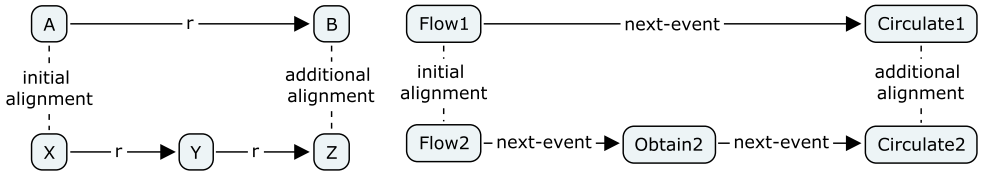
Pattern 1. **(simple alignment)** There are triples (A r B) in G1 and (X r Y) in G2 such that B subsumes Y (or *vice versa*). Then, B is mapped with Y.



This pattern can align, for example, (Engine1 has-part Piston1) with (Gasoline-Engine2 has-part Piston2) if the system can infer Engine and Gasoline-Engine are taxonomically related.

Patterns 2 through 6 resolve several types of granularity mismatches.

Pattern 2. **(transitivity-based alignment)** There are triples (A r B) in G1 and (X r Y) (Y r Z) in G2 such that B subsumes Z (or *vice versa*) and r is a transitive relation such as causes, next-event. Then, B is mapped with Z.



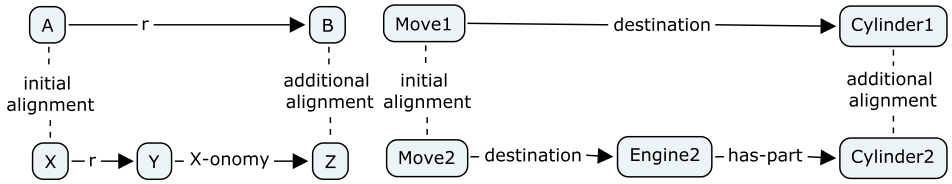
This pattern resolves a granularity mismatch caused by a transitive relation, as shown in the example. The top representation expresses *"The blood flows to the lung and then circulates around the body."* and the bottom one *"The blood flows to the lung to pick up oxygen and then circulates around the body"* [5]

The following pattern uses a relation called *X-onomy*, which is a general

[5]The representation omits the parts that are not related to the pattern.

relation that includes all relations that involve hierarchy such as *has-part*, *has-region* and *sub-event* (partonomy), *isa* (taxonomy).

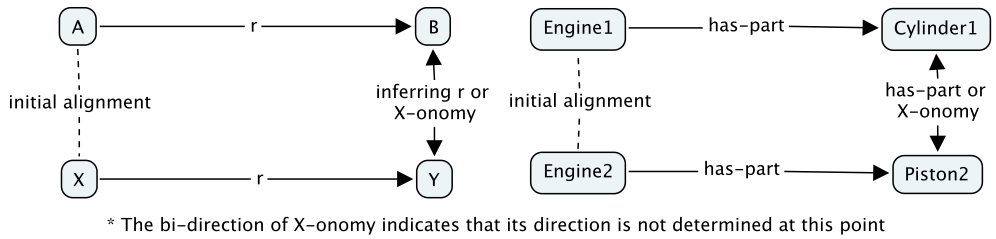
Pattern 3. **(Co-reference across a granularity shift)** There are (A r B) in G1 and (X r Y) (Y X-onomy Z) in G2 such that B subsumes Z (or *vice versa*). Then, B is mapped with Z.



This pattern handles a case that two expressions reference the same entity or the same event at different granularities. For example, two representations, *"The engine takes in gasoline"* and *"The cylinder in the engine takes in gasoline"*, co-references at the different granularity the location which gasoline is taken into. Notice that the pattern 3 can align (Move1 destination Cylinder1) with (Move2 destination Engine2) (Engine2 has-part Cylinder2). In this case, X-onomy is a *has-part* relation.

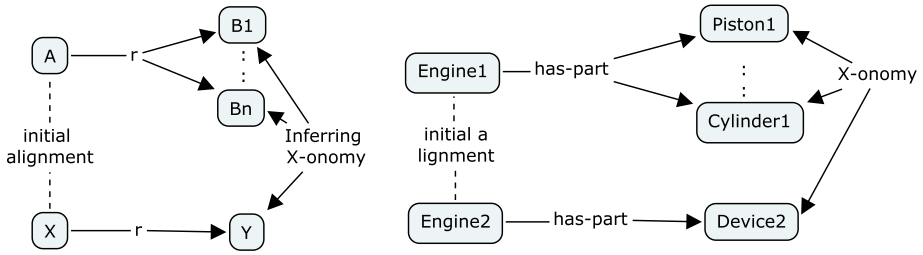
Patterns 4 through 6 introduce additional triples without which the alignment would fail.

Pattern 4. **(Abductive transfer-thru alignment)** There are triples (A r B) in G1 and (X r Y) in G2 such that B does not subsume Y (or *vice versa*). Then, X-onomy can be abduced between B and Y. Additionally, if r is a transitive relation, r can be abduced between B and Y, too. (In the implementation of Kleo, we abduce *related-to*).



This pattern is an abductive version of pattern 2 and pattern 3. The example shows that, for *"An engine has cylinders"* and *"An engine has pistons"*, this pattern makes an inference that *"cylinders"* and *"pistons"* are in X-onomy relation – in this case, (Cylinder1 encloses Piston2)).

Pattern 5. **(Generalization-based alignment)** There are (A r B_1) .. (A r B_n) in G1 and (X r Y) in G2 such that each of B_1, ..., B_n does not subsumes Y (or *vice versa*). Then, X-onomy is abduced between B_i and Y for i = 1,2, ..., n.

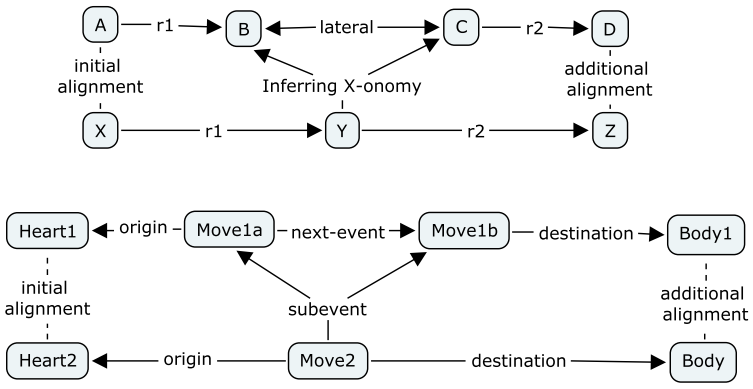


This pattern handles a case that several pieces of similar information in a fine-grained representation are generalized together to form a coarse-grained representation. For example, given two representations, (Engine1 has-part Cylinder1) (Engine1 has-part Piston1) (*"An engine has a cylinder and a piston"*) and (Engine2 has-part Device2)(*"An engine consists of parts"*), this

pattern makes an inference that Cylinder1 and Piston1 are in a X-onomy relationship with Device2.

The following pattern uses a relation called *lateral relation*, which is a general relation that connects two consecutive things whether ENTITY or EVENT (e.g. *next-event*, *beside*).

Pattern 6. **(Abstraction-based alignment)** There are triples (A r1 B) (B lateral-relation C) (C r2 D) in G1 and (X r1 Y) (Y r2 Z) in G2. Then D is mapped with Z, and two X-onomy relations are abduced between Y and B and between Z and C.



This pattern handles a case that an aggregation of small things, whether ENTITIES or EVENTS, is viewed as one thing. The above example (a simplified form of Figure 2.7) shows that this pattern infers that Move2 is a superevent of Move1a and Move1b.

All these patterns are inherently uncertain; That is, a pattern could make a wrong alignment even if its precondition holds true (especially, pat-

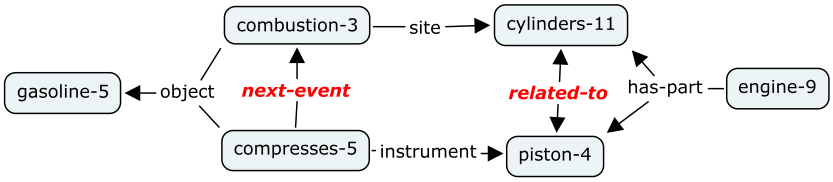


Figure 3.6: The stitched representation

tern 4-6). Part of our future research is to deal with the uncertain nature of knowledge integration.

3.2.2 Sentence-to-Sentence Knowledge Integration

SKI produces a knowledge representation (text-KR) of all the content extracted from a single text. Initially, the text-KR consists of only the representation of the first sentence. As Kleo reads each subsequent sentence, the representation of the sentence is integrated with its text-KR. The goal of SKI is to relate entities and events mentioned in a text coherently. To assess the contribution of SKI – i.e., its impact on reducing knowledge fragmentation, we will measure the improvement it causes in the density (connectivity) of the graphical representations of its interpretation of texts

SKI consists of two steps: *Stitch* and *Elaborate*. *Stitch* aligns the representation of each sentence with the text-KR, and *Elaborate* make explicit its implied content.

3.2.2.1 Stitch

Stitch aligns two semantic representations by graph matching, as ex-

plained in Section 3.2.1. Beyond resolving some direct anaphoric references among the sentences, *stitch* can align graph representations that are semantically identical. *Stitch* also performs a simple mapping (similar to [12]) [6] from cue phrases, such as *"because", "and so"*(causal), and *"then", "after"*(temporal), to semantic relations between the consecutive sentences.

The matcher combines S1 and S2 in Figure 3.3 as follows:

- (`engine-8`, `engine-2`) and (`gasoline-4`, `gasoline-6`) are selected as pairs of seed nodes because their type, DEVICE and LIQUID-SUBSTANCE, are entities and they derive from the same word classes.
- The matcher produces a new triple, (cylinder-10 related-to piston-3), from a pair of seed nodes, (`engine-8`, `engine-2`), according to the extension pattern 4.
- The matcher produces a new triple, (combustion-1 next-event compresses-4), from a cue word, *"then"*.

3.2.2.2 Elaborate

Elaborate augments the representation produced by *Stitch* by adding relevant background knowledge. This step relates together two or more nodes in the graphical representation with semantic paths, which could resolve in-

[6] It is not a focus of our reserach to identify such mappings, which have been extensively studied in the research of the discourse relations. Any state-of-art software which identifies discourse relations, such as [94], can be plugged in Kleo.

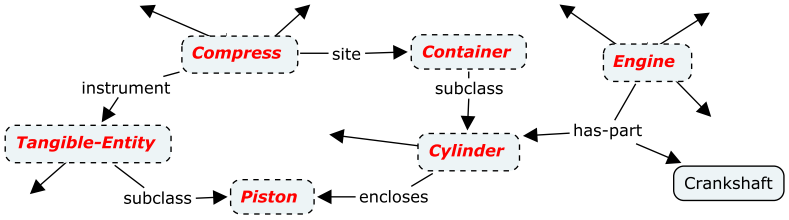


Figure 3.7: The example knowledge base

direct anaphora [47] or add temporal/causal relations among the events mentioned in the text.

The knowledge base from which Kleo draws background information consists of two sources. One is the Component Library which provides rich representations of general events and entities. The other source is relations that Kleo learned by previously reading other texts on the same topic.

The background information is retrieved by spreading activation search through both sources. Specifically, the activation starts from the concepts referenced in the representations produced by *Stitch*. For example, suppose that the knowledge base - i.e. the Component Library or pre-learned knowledge - contains the representation shown in Figure 3.7. For the representation in Figure 3.6, activation starts from ENGINE, CYLINDER, PISTON, and COMPRESS which are the types of `engine-9`, `cylinders-11`, `piston-4` and `compresses-5`. Currently, we use a simple termination condition whereby activation stops when it meets another activation (success) or it travels a certain distance (failure). In our example, the italicized parts - the region explored

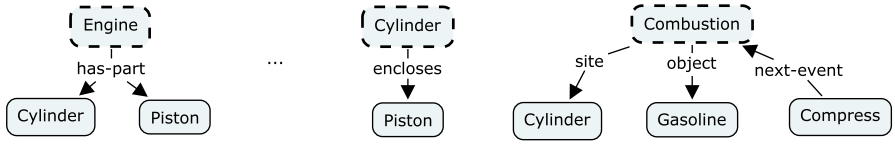


Figure 3.8: K-units produced from the representation in Figure 3.5 by partitioning. The dotted nodes are roots.

by the activation (expressing *"The compression occurs inside the cylinder"* and *"The cylinder encloses the piston"*) - are returned as background information and aligned with the representation in Figure 3.6. Figure 3.5 shows the resulting representation.

3.2.3 Text-to-Text Knowledge Integration

The goal of TKI is to combine knowledge extracted from multiple texts into a single coherent knowledge base. Initially, the knowledge base consists of only the representation of the first text. As Kleo reads each subsequent text, the representation of the text is integrated with the knowledge base. Typically, the texts have overlapping content. TKI attempts to identify the points of overlap, which is challenging because the texts differ in their presentation. For example, the texts present overlapping content, but at different levels of granularity.

The graph matcher (in Section 3.2.1) is the main component of TKI. Before it can be applied, however, an efficiency concern should be addressed because TKI attempts to match graphs that are large (Contrast this with SKI, which attempts to integrate graphs that represent single sentences). Our ap-

proach is to partition the graphs into a set of coherent subgraphs, that we will call K-units. Figure 3.8 shows examples of K-units for ENGINE, CYLINDER, and COMBUSTION.

A K-unit has a root node which is universally quantified. The other nodes in the K-unit are existentially quantified in the scope of the root node. For example, the CYLINDER K-unit in Figure 3.8 represents $\forall x.Cylinder(x) \rightarrow \exists y.Piston(y) \wedge encloses(x, y)$. Notice that the K-units are implicitly associated to one another because the root nodes are universally quantified. For example, CYLINDER in the ENGINE K-unit in Figure 3.8 is connected to the root node of the CYLINDER K-unit.

TKI partitions learned knowledge into K-units with the following procedure: Given a text-KR from SKI, TKI chooses nodes that can serve as root nodes - typically, key concepts in the domain of the texts. The heuristic used by TKI is selecting a node whose input words frequently appear in the texts or a node that is an instance of an EVENT because frequently occurring words or events are generally important concepts. Once the root nodes are identified, TKI performs spreading activation from each root node until the activation arrives at other root nodes. The region explored by the activation becomes a K-unit for the root node. The K-units in Figure 3.8 result from partitioning the representation in Figure 3.5.

For each new K-unit, Kleo retrieves a relevant K-unit from the knowledge base. Two K-units are relevant in the two following cases: if their roots are instances of the same entity; if their root nodes are events and their case

roles are taxonomically related. The two relevant K-units are aligned by the graph matcher and the new integrated K-unit is stored in the knowledge base.

3.3 Evaluation

We evaluate the contribution of knowledge integration by comparing Kleo with Mobius, two systems that differ primarily in their knowledge integration capabilities [7]. Specifically, they differ in three ways: (1) Kleo's graph matcher handles mismatches in granularity, while Mobius's does not (i.e. Kleo uses patterns 1 - 6, described in Section 3.2.1, whereas Mobius uses only pattern 1). (2) During knowledge integration, Kleo uses both Component Library and knowledge derived from reading previous texts. Mobius uses only Component Library. (3) To improve efficiency, Kleo uses partitioning (see Section 3.2.3), but Mobius does not.

Both systems read 25 texts in two domains: the blood circulation in the human heart and the cycle in the internal-combustion engine. Each text consisted of a paragraph drawn from a variety of sources – encyclopedia, web pages, textbooks – and were roughly at the same complexity as the Wikipedia articles on the topics. On average, the texts contained 5.67 sentences, and the sentences averaged 16 words in length.

[7] The original Mobius system used BBN's Serif parser [104]; but, for the evaluation purpose, we replaced the parser with the Stanford parser used in Kleo.

3.3.1 Evaluation of SKI

We evaluate the two main features of SKI: (1) the ability to align representations, even when they differ in their level of granularity. This is performed by pattern rules 2 through 6 in the graph matcher (see Section 3.2.1) (2) the use of background knowledge learned by reading previous texts (see *Elaborate* in Section 3.2.2.2).

To measure the contribution of each feature, we compared 4 systems: Kleo (uses both features), Mobius+GM (uses only the extension rules), Mobius+KB (uses only the pre-learned knowledge), Mobius (uses neither). The graph matcher used in Mobius and Mobius+KB is the same as the one in Kleo except that it uses only rule pattern 1, and none of the rules that handle granularity mismatches.

The main purpose of SKI is to reduce the fragmentation of knowledge among the sentence-level representations. To measure the contribution of SKI, we use a metric, *density-improvement*. This metric is based on the *density* of the graph, which measures the connectivity of a graphical representation:

$$density = \frac{\text{number of edges}}{\text{number of edges in a fully connected graph of n nodes}} = \frac{e}{\binom{n}{2}}$$

where n is the number of nodes in the graph and e is the number of edges. The maximum of *density* is 1 in the case that the graph is fully connected.

Density-improvement is defined as

$$\frac{\text{density of the graph after SKI}}{\text{density of the graph before SKI}}$$

This metric measures SKI's contribution to increase the density of the graphical representation. This metric alone, however, is insufficient because it considers only the structural parts of the graph. For example, if all nodes in the representation are merged into a single node, the operation would maximize *density-improvement* but would be wrong for most cases. To compensate *density-improvement*, we hired a third party to measure *operation accuracy*, the ratio of the correct knowledge integration operations. Specifically, each knowledge integration operation is scored in the following way and then the scores were averaged:

1	if an operation coherently combines knowledge snippets
.5	if an operation is partially justified
0	if an operation is not justified.

There are two cases in which the operations are partially justified: the background knowledge used by *Elaborate* could be only partially correct; merging two nodes could be only partially correct.[8]

Evaluation Result

[8]For example, let's assume that node3 should be merged with node1 but not with node2 and that knowledge integration merges node1 and node2 to produce node4 and then merges node4 with node3. Merging of node4 with node3 is partially justified because node1 should be merged with node3 but node2 should not.

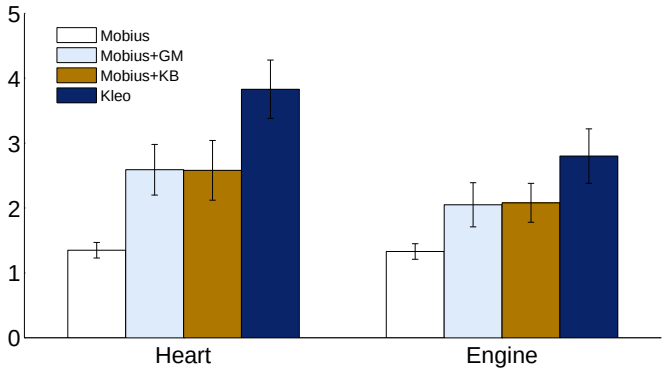


Figure 3.9: The average density-improvement over 25 texts in each domain. The experiment was repeated 10 times with the different order of reading. The lines represent the standard deviation. The difference between Mobius and Mobius+GM/Mobius+KB and between Mobius+GM/Mobius+KB is statistically significant in both domains ($p<.05$ for the two tail z-test).

Figure 3.9 shows the results. SKI improved the density of the graph of learned knowledge by a factor of 2.5-3.5 (the *density-improvement* of Kleo is 2.5-3.5 times higher than the *density-improvement* of Mobius). The two main features of SKI – handling granularity mismatches and using background knowledge from previous reading – contributed equally to this improvement (the *density-improvement* of Mobius+GM and Mobius+KB are about equal).

Further, we investigated why SKI contributed more in the heart domain than in the engine domain. We found that the major reason for this difference is that Kleo drew more background knowledge for the texts about the heart because they have more overlapping content, as measured by the number of words in common across the texts.

The average operation accuracy is .56, indicating that roughly half of

the operations contribute to the goal of coherently combining information. However, when we considered only the cases where the input knowledge snippets are correctly formulated, the score is .81.

3.3.2 Failure Analysis on SKI

To prioritize the future work on knowledge integration, we analyzed the failure cases of SKI during the experiment in Section 3.3.1. We randomly chose 15 texts used in the experiment, and closely examined the following knowledge integration operations – merging two knowledge snippets and inferring additional semantic relations. The following shows the categories of the failures with their frequencies.

1. **Incorrect semantic representations from previous reading (36%)** During knowledge integration, Kleo draws upon knowledge acquired from reading previous texts. In errors of this type, the previously acquired knowledge was incorrect, typically due to errors in the NL component.

2. **Irrelevant background knowledge (18%)** In errors of this type, Kleo pulled in previously acquired knowledge that is irrelevant. For example, Kleo pulled in information about *a jet engine* for a text about *gasoline engine*. To avoid this error, Kleo should consider context when drawing information from the background knowledge base.

3. **Overly general background knowledge (18%)** In errors of this type, Kleo pulled in knowledge that is overly general and vague. For example,

Kleo pulled in the triple (Engine related-to Piston), which fails to specify the relationship.

4. **Inappropriate application of Heuristic1 in Section 3.2.1 (14%)** In errors of this type, Heuristic1 joined two entities that share a common head word, but are otherwise dissimilar. For example, Kleo inappropriately joined *"right atrium"* with *"left atrium"*. To avoid this type of mistake, Kleo should consider features derived from modifying phrases.

5. **Language interpretation failure (8%)** In errors of this type, Kleo makes incorrect knowledge-integration decisions because the NL component misread a text. For example, mapping a word to an inappropriate concept can cause errors in graph matching.

6. **Mishandling transient properties (5%)** In errors of this type, Kleo fails to recognize that a property is transient, causing inappropriate matches. For example, Kleo incorrectly matched *"oxygen-rich blood"* with *"oxygen-poor blood"*

Kleo significantly suffers from the brittle NLP as shown in case (1) and (5) (about 45% in total). In Chapter 4, we will show how knowledge integration can help improve language interpretation.

3.3.3 Evaluation of TKI

The main purpose of TKI is to combine information learned across multiple texts. In this evaluation, we use the metric of knowledge base size – the

number of triples in the final knowledge base of information learned by reading a corpus of texts. Because TKI attempts to find overlapping content, to avoid redundancy and improve coherence, the more TKI identifies and merges the overlapping contents the smaller the resulting knowledge base will be.

Specifically, we evaluate the contribution of the graph matcher by comparing three systems: Kleo (uses all the pattern rules described in Section 3.2.1), Mobius (uses only pattern rule 1, not rules 2 through 6, which are intended to handle granularity mismatches), and a baseline (simply appends knowledge structures with no attempt to match them).

Figure 3.10 shows the result. The x-axis is the total number of input triples presented to TKI and the y-axis is the total number of output triples produced by TKI (i.e. the total number of triples in the final knowledge base). In both domains, TKI discovered a significant amount of overlapping content across the texts – the knowledge base built by Kleo was 30% smaller than the one built by the baseline. The graph-matcher rules that handle granularity mismatches account for about 45% of this contribution – the knowledge base built by Kleo was 18% smaller than the one built by Mobius.

We also evaluated the contribution of partitioning. The purpose of partitioning is to dampen the increase in the cost of graph matching as the knowledge base grows. We compare two systems: Kleo and Kleo-without-Partitioning (i.e. Kleo without the partitioning capability).

Figure 3.11 shows the results. The time required to read a text by

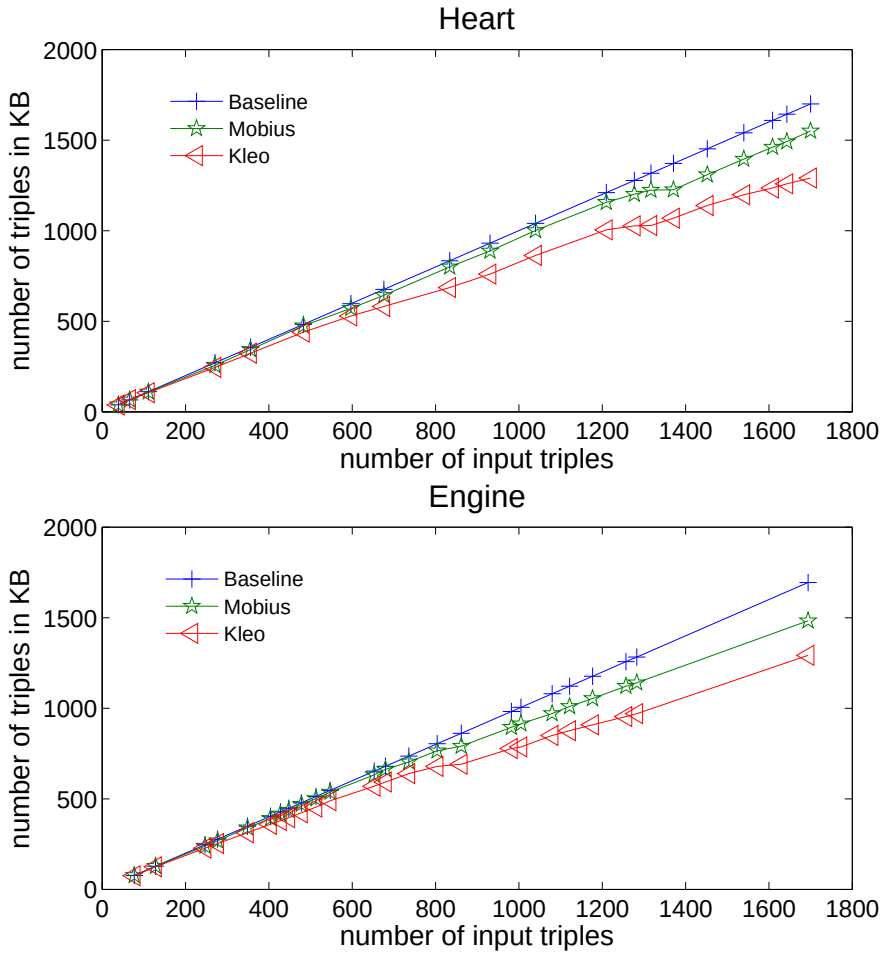


Figure 3.10: Increase in knowledge-base size with presentation of triples to TKI

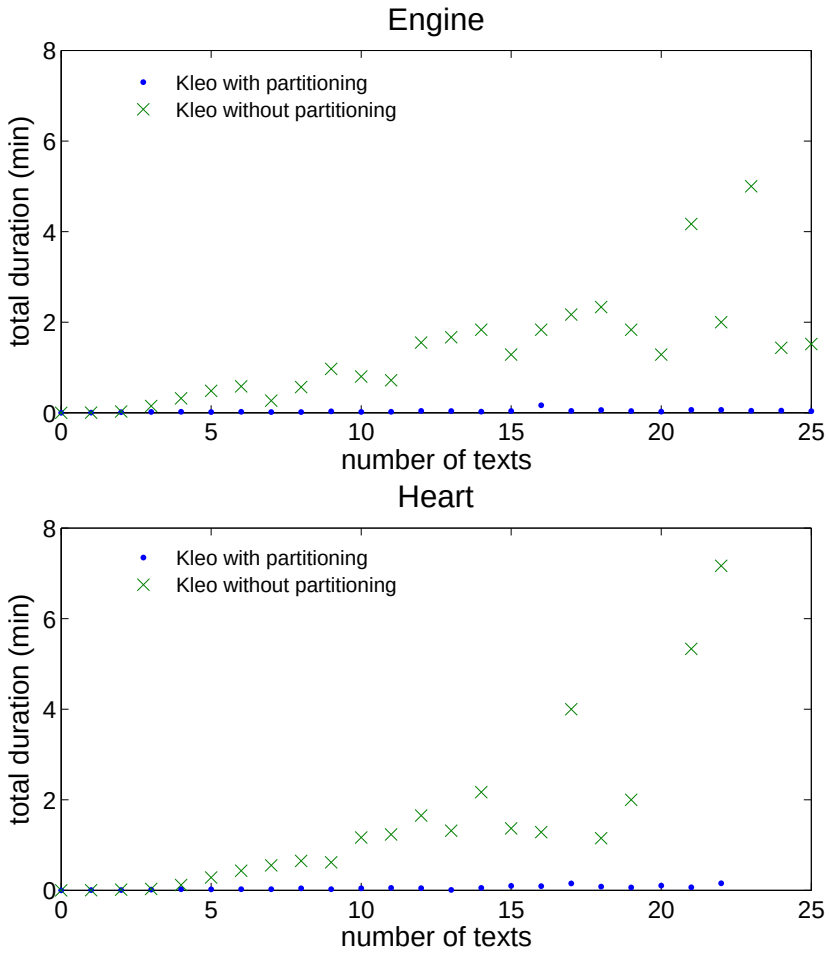


Figure 3.11: With partitioning, the run time of updating a knowledge base is almost negligible

patterns	frequency(%)	patterns	frequency(%)
pattern1	52.4	pattern2	0.3
pattern3	8.2	pattern4	31.9
pattern5	7.2	pattern6	0

Table 3.3: The frequency of invocation of the extension pattern rules

Kleo-without-Partitioning grows rapidly with the size of the knowledge base (as measured here by the number of texts read so far). In contrast, the time required by Kleo remains almost constant. Partitioning seems crucial for the scalability of knowledge integration.

Finally, we measured the number of times in which the extension pattern rules are invoked during reading the 25 texts in two domains during both SKI and TKI. Table 3.3 shows the result. It shows that the rules for resolving granularity mismatches (pattern 2 $\sim$ 6) make a significant contribution (approximately 50% of total invocation) and that Pattern4 is used most whereas Pattern2 and 6 are rarely used.

3.4 Related Work

We present related work on three issues important to knowledge integration: handling granularity in other AI tasks (Section 3.4.1), flexibly aligning knowledge snippets beyond their surface forms (Section 3.4.2 and 3.4.4), and inferring unspecified information in texts (Section 4.4.2).

3.4.1 Handling Granularity in Reasoning

Handling granularity is one of the central problems in building intelligent systems. For example, an agent should be able to identify the appropriate level of granularity when perceiving the outside world or using background knowledge for its reasoning. For this reason, granularity has been extensively studied in many fields of AI.

Ontology construction requires choosing an appropriate granularity in modeling domain knowledge. Bittner and Smith [15] propose a formal theory of granular partitions (i.e., ways of dividing reality) and applies it to the construction of spatio-temporal ontologies. Ontology alignment, as in our knowledge integration, should resolve the granularity mismatches among the individual ontologies [44].

Granular computing is an emerging computational framework concerned with developing methods for analyzing data at multiple granularities and manipulating information granules [147]. For example, in the analysis of satellite images, meaningful information can be identified at different granularities – for example, at the macro level (e.g., the large-scale distribution of the clouds) or the micro level (e.g., the streets in a city). One may shift through the different granularities to identify the interesting patterns among them. Granular computing has primarily focused on numerical representations of data rather than symbolic representations, which is the aim of our task.

In NLP, Hobbs proposes a theory of granularity to model several types

of reasoning related to granularity [67] and then uses this theory to decompose complex lexical knowledge into simple primitives [68]. Mani [91] also develops a theory of granularity to analyze polysemy and underspecification.

3.4.2 Semantic Decomposition

Semantic decomposition is concerned with breaking down a conceptual meaning into primitive elements and their semantic relations. This work can be useful for flexible semantic matching, specifically to address *lexical compression*, as explained in Section 2.3.3.1, by using more canonical ways to represent the meaning. Unfortunately, a large-scale computational resource that provides these semantic decomposition data has not yet been developed.

Conceptual Dependency [126] is the early work in AI that proposes 11 primitive concepts (called primitive acts) and semantic relations, such as the case roles. The Component Library [11] encodes knowledge about the domain-independent concepts and then extends the concepts to represent complex conceptual knowledge. Similarly, Hobbs [68] proposes an abstract theory to logically define abstract concepts (which correspond to the high-level synsets in WordNet) and then extends the axioms defined in the abstract theory to represent more complex concepts (the specific synsets). His abstract theory includes knowledge about the system, figure-ground relation, scale, change of state, causality, and goal-directed systems.

In Lexical Semantics, Jackendoff proposes Lexical Conceptual Structure to decompose the meaning of the lexical item. For example, *give* is decomposed

formal	properties that distinguish the object in a larger domain
constitutive	the physical properties or the parts
telic	the purpose or function of the object
agentive	object's origins that make the object come into being

```
formal :  artifact tool
constitutive :  metal blade, handle, etc.
telic :  cutting
agentive :  making
```

Table 3.4: The factors in the qualia structure and an example of the qualia structure for *scissor*.

into [CAUSE [x, [GO [y, [TO (z)]]], which means that an agent(x) moves an object(y) to a destination(z). Pustejovsky proposes the Qualia Structure in his Generative Lexicon framework to characterize lexical items using several factors. Table 3.4 describes these factors along with the example of the qualia structure for *scissor*.

3.4.3 Analogical Reasoning

Analogical reasoning allows humans to understand a new domain (called the "target domain") by transferring information from a base domain to the target domain. For example, knowledge about a water circuit system (in which a water pump pumps water to flow through a pipe) can be used to understand an electrical circuit system (in which a battery "pushes" electrons through a wire) because of their analogical similarity. This reasoning method is a primary way in which humans create new concepts and knowledge.

One of the seminal works in analogical reasoning research is the struc-

ture mapping theory [46]. This theory argues that analogical relationships arise when two pieces of knowledge (from two different domains) are structurally similar – for example, the water circuit and the electrical circuit are described as having similar structure – X causes Y to move through Z. To identify these structural mappings, the theory proposes an algorithm that performs graph matching subject to several constraints that the analogical mappings should satisfy. It is interesting future work to incorporate these constraints in our flexible matcher, because they might allow the system to relate knowledge from different domains – e.g., using background knowledge from different domains in the *elaborate* step.

3.4.4 Paraphrase Discovery

Paraphrase Discovery [88], actively researched in the question-answering community, aims to find all possible paraphrased forms, given a particular sentence. This work is important for question-answering because a question often uses phrases formulated differently than the one with the answer. Paraphrase patterns can be of further use for Kleo because knowledge integration should perform flexible semantic matching on natural sentences.

3.4.5 Underspecification

Texts often underspecify the meaning in the underlying representations to facilitate efficient communication. Various types of underspecification exist, one of which is Polysemy, a lexical-level underspecification in which the

same word is used to represent different meanings. For example, Metonymy, a primary device for polysemy, allows a thing to be referred not only by its own name, but also by a closely related name (e.g., Using *Washington* to represent the US government, as well as the geographical city). Noun-noun compounds also underspecify the relation between two nouns. Presupposition is underspecified contextual knowledge that people assume in their communications. For example, *"John wants to graduate this year"* presupposes that John is currently a student and has not yet graduated.

To reveal underspecified knowledge, the system needs background knowledge. Kleo uses knowledge acquired from previous reading to reveal unspecified information in texts. Traditionally, three approaches have been used to build a large-scale background knowledge base for text understanding: manual construction, automated construction, and construction by volunteers.

The manual approach generally produces a knowledge base that is more computationally useful than the other two methods because the manual encoding and organization of the knowledge representations outperform the other two approaches in terms of accuracy and coherence of the representations. However, this approach is expensive as it requires costly expertise, construction time and human labor. Example knowledge bases constructed by this approach are CYC [85] (world knowledge), FrameNet [51] (knowledge about typical situations), Wordnet [103] and Verbnet [129] (lexical knowledge), and the Component Library [11] (knowledge about domain-independent concepts).

Unlike the manual approach, an automated approach is less costly, but

often produces poorly formulated knowledge representations. Because of this poor quality, few knowledge bases in this category are widely used for machine reading. Example knowledge bases constructed using an automated approach are Prismatic [48] and Knight [127] (common sense knowledge), VerbOcean [32] (the relations between two verbs), and DIRT [88] (paraphrase patterns).

The most recent approach, using construction by volunteers, engages the general public to build the knowledge base together. Using this approach, the central system receives small pieces of knowledge from a large number of volunteers and then combines these pieces into a coherent knowledge base. This approach addresses the disadvantages of the two other approaches: it reduces the building cost by involving the general public and improves the quality of the semantic representations by requiring manual encoding. Despite these advantages, however, there has yet to be a widely used knowledge base built by this approach; coordinating many volunteers and coherently integrating the small pieces of contributed knowledge remains challenging. Example knowledge bases constructed by this approach are OpenMind [130] and Learner [31] (common sense knowledge).

To build a background knowledge base for text understanding remains one of the greatest challenges in AI.

Chapter 4

Application of Knowledge Integration to Text Interpretation

Despite the significant progress in Natural Language Processing, full interpretation of a sentence is still a major bottleneck in machine reading. Our pilot study on the NL component in Kleo (see Table 3.1) confirms this brittleness, particularly the difficulty of semantic interpretation (such as the tasks of assigning types and semantic relations).

One reason for this brittleness stems from the pipeline architecture, which has been widely used in machine reading systems, including Mobius and Kleo. As (a) in Figure 4.1 shows, the pipeline architecture connects the components (e.g., for parsing, word sense disambiguation, and semantic relation assignment) serially, and each component passes a single solution to the next component in line. The problem with the pipeline architecture is that each component is forced to choose a single interpretation, even in the presence of significant ambiguity and insufficient evidence to resolve it. Furthermore, because errors can magnify when a component commits to a wrong interpretation, downstream components are misled into making more errors.

One approach to addressing this problem, albeit naïve, is for the system

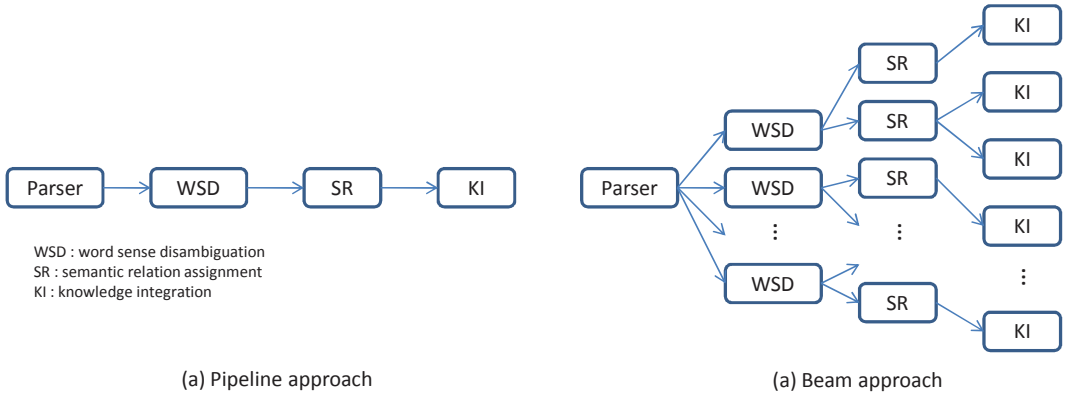


Figure 4.1: (a) The traditional pipeline approach; (b) the beam approach to maintain multiple candidate interpretations

to maintain multiple complete candidate interpretations using, for example, a beam of the n-best interpretations (see (b) in Figure 4.1). While this approach can delay ambiguity resolution, several problems remain. First, the number of interpretations is generally very large. Because of the limited size of the beam, the system may still discard correct interpretations before benefiting from knowledge integration. Second, the candidate interpretations generally do not represent the dependencies among the interpretations. For example, there may be multiple candidate word senses and semantic roles for a given sentence, but sense alternatives might depend on role selection (and vice versa). The set of reasonable interpretations may be a subset of all combinations. Finally, maintaining distinct interpretations does not contribute to addressing the problem of combining evidence to narrow down alternatives and ultimately select the best interpretation.

In this chapter, we propose an approach to address these three prob-

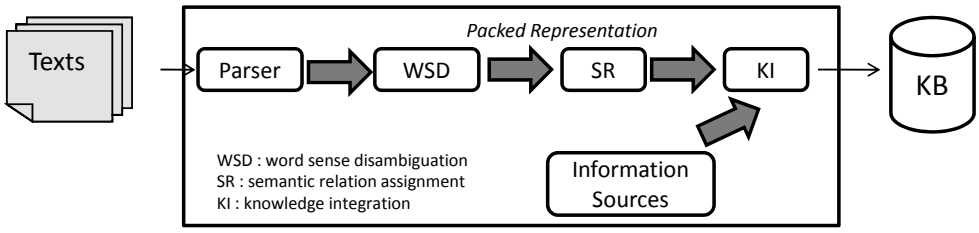


Figure 4.2: Our system architecture based on the packed representation

lems. In our approach, the system postpones committing to an interpretation of a text by representing ambiguities and the dependencies among them. Our approach may include combinatorial growth in the set of alternative interpretations, but they are represented only intensionally using the packed representation, i.e., a single representation that succinctly maintains alternatives while avoiding enumerating them.

Figure 4.2 shows our proposed architecture, which is based on the packed representation. The system delays ambiguity resolution by maintaining multiple candidate interpretations using the packed representation. Then, knowledge integration – the last step of the pipeline – chooses the overall best interpretation from the packed representation by aggregating evidence from a variety of information sources.

Based on this architectural framework, we explore three sources of information that could be useful for ambiguity resolution in knowledge integration. The information sources are redundancy across multiple texts, OntoNotes (a

semantically annotated corpus [1]) and Prismatic (a knowledge resource automatically constructed from texts). These information sources can be applied at different points in the pipeline. Except Prismatic, note that redundancy and OntoNotes can only be exploited downstream in the pipeline; redundancy, an inter-sentential property, can only be exploited in knowledge integration; and, OntoNotes, with no syntactic information, cannot directly benefit the parser. This shows the benefit of using the packed representation to delay ambiguity resolution – it allows the system to resolve ambiguities occurring upstream (e.g., parsing ambiguities) using the information available only downstream.

Our experiments show that for two of these sources – redundancy and OntoNotes – our approach produces significantly better semantic representations than the traditional pipeline approach because knowledge integration exploits these information sources effectively by using the packed representation. Our initial method for using Prismatic is found to be partially effective.

This chapter is organized as follows. In Section 4.1 and 4.2, we present two packed representation schemes that can succinctly represent a myriad of candidate graphical semantic representations. Sections 4.1 through 4.3 introduce the three sources of information: redundancy across multiple sentences (Section 4.1), OntoNotes (Section 4.2), and Prismatic (Section 4.3). The sections also present the knowledge integration algorithms to show how the information is used for disambiguating the packed representation. Finally,

[1] OntoNotes also contains the syntactic annotations; for our study, however, we use only its semantic annotations, word sense annotations, and semantic role annotations.

Section 4.4 presents the related work.

4.1 Using Redundancy across Multiple Texts

Within a corpus of texts on the same topic, the texts could express the same meaning in different surface forms; consequently they are ambiguous in different ways. When the system interprets these redundant texts, it may use the interpretation of a text to provide context that help inform the interpretation of the others. Related fields, such as Information Extraction, exploit textual redundancy to good effect [39], and perhaps text understanding can as well.

Figure 4.3 illustrates the benefit of using redundancy in the interpretation of the two following redundant sentences, each of which is ambiguous to interpret.

S1: An engine ignites gasoline with its spark plug.
S2: The engine's spark plug ignites gasoline.

In S1, an ambiguity arises in attaching the prepositional phrase, *with its spark-plug*. It can attach either to *ignites* (correct attachment) or to *gasoline* (incorrect attachment). The different interpretations result in two different semantic representations as shown in A1 and A2 in Figure 4.3. In S2, the syntactic subject relationship is ambiguous to interpret. In general, the syntactic subject can map to different semantic relations depending on the context. For example, it can map to *agent* (as in "John throws the ball"), *instrument* (our

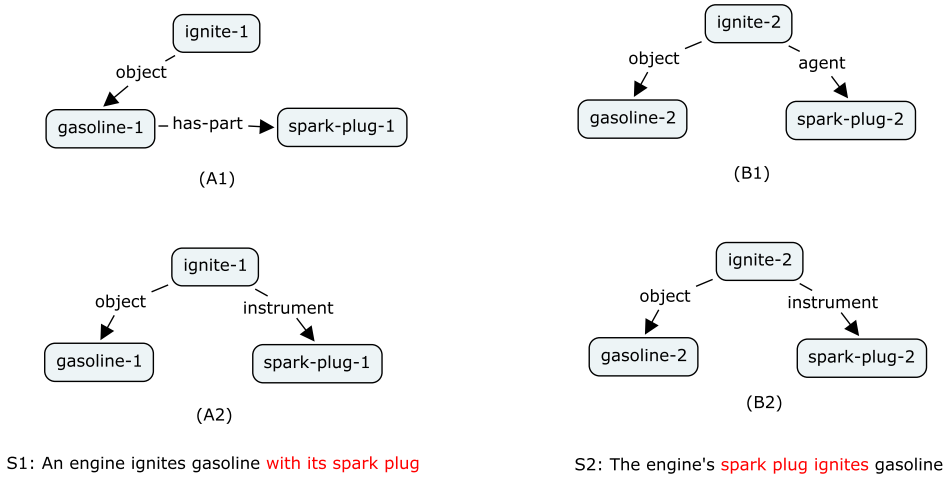


Figure 4.3: Two candidate interpretations for each of S1 and S2. The interpretations, A2 and B2, which are identical, are more likely to be correct than the others because the sentences express the same meaning.

example, S2), *recipient* (as in "John received the book"), or *destination* (as in "The engine takes in the gasoline").

The independent resolution of these ambiguities may be difficult but jointly resolving them may ease the problem due to their redundancy. Because the two sentences encode the same meaning, their meaning representations should be identical and therefore, the two candidate semantic representations, A2 and B2, in Figure 4.3 (which are identical), should be preferred over the others. This example also shows the advantage of maintaining multiple candidate interpretations, the capability provided by the packed representation. Without this capability, the system might be forced to choose the wrong representations for both sentences before exploiting the redundancy.

In this section, we present one packed representation scheme and a

knowledge integration algorithm that disambiguates the packed representation based on redundancy (Section 4.1.1 and 4.1.2). We also present a prototype system, Ally, that implements our approach (Section 4.1.3). We evaluate our approach by comparing the accuracy of two reading systems: a baseline system that commits to its best interpretation after each sentence, and Ally, which uses a packed representation and our algorithm to maintain all possible interpretations until further reading enables it to prune. For this initial proof of concept, we use a small corpus of redundant texts. The results indicate that our approach improves the quality of text interpretation by preventing aggressive pruning while avoiding combinatorial explosion (Section 4.1.4 through 4.1.6).

4.1.1 Packed Representation

Multiple alternative semantic interpretations for a sentence can be captured with a single packed representation in which ambiguities are represented as local alternatives. Because the candidate semantic representations are often structurally similar, a packed representation can significantly compress the representation of alternative interpretations.

Figure 4.4 shows the packed representation of alternative interpretations of S1 (PG1). The different types of ambiguity captured by the packed representation are as follows.

1. **Type ambiguity** Ambiguity in the assignment of a type for a word. In PR1, the node `engine-2a` corresponds to the word *"engine"* in S1. Its annotation [LIVING-ENTITY .3 | DEVICE .7] says that the word may

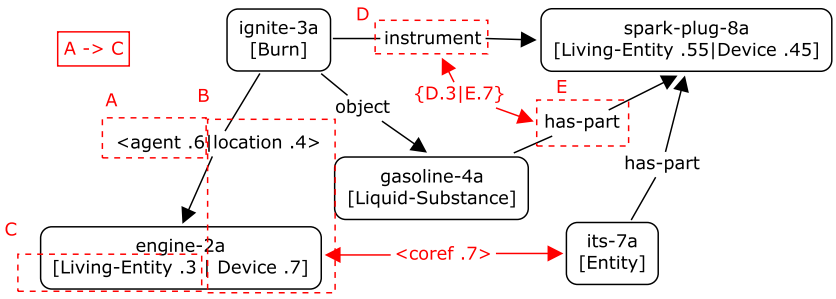


Figure 4.4: The packed representation for S1 (PR1)

map to either LIVING-ENTITY (probability 0.3) or DEVICE (probability 0.7). The packed representation does not presume a particular uncertainty formalism. Any formalism, (e.g., Dempster-Shafer theory [113] or Markov Logic Networks [124]) could be used.

2. **Relational ambiguity** Ambiguity in the assignment of semantic relation between nodes. In PR1, the edge label <agent .6 | location .4> from `ignite-3a` to `engine-2a` represents that the engine is either *agent* or *location* of the ignition.

3. **Structural ambiguity** The packed representation also captures structural alternatives. In PR1, edges D and E are alternatives corresponding to the different prepositional phrase attachments for *"with its spark plug"* (to `ignite-3a` or `gasoline-4a`). The annotation {D .3 | E .7} represents that the choices are mutually exclusive with probabilities of 0.3 and 0.7.

4. **Co-reference ambiguity** Co-reference of nodes in a packed representation is captured using a "co-reference" edge. In PR1, the edge labeled

<coref .7> represents that the probability of `engine-2a` and `its-7a` being co-referent is .7.

In addition to storing ambiguities explicitly, the packed representation also captures dependencies among alternatives.

5. **Simple dependency** The existence of one element in the graph depends on the existence of another element. If subsequent evidence suggests that an element is incorrect, its dependents should be pruned. For example, the dependency, A $\rightarrow$ C, means that if LIVING-ENTITY is ultimately rejected as the type for `engine-2a`, the agent relation should be pruned.

6. **Mutual dependency.** Elements of a mutual dependency set are mutually confirming. If enough evidence accrues to confirm or reject an element, other elements in the set should also be confirmed or rejected. In the example, the box labeled B says that the two elements, (engine-2a type DEVICE) and (ignite-3a location engine-2a), should both be confirmed or pruned when either of them is confirmed or pruned.

Similar to belief maintenance systems [37], these constraints enable the system to adjust each candidate's confidence score in response to changes to other interpretations. Formally, the packed representation is a structure consisting of (a) *semantic triples* – e.g., (ignite-3a type BURN), (b) *macros* – e.g., the symbol A refers to (ignite-3a agent engine-2a), and (c) *constraints* – e.g., A depends on C.

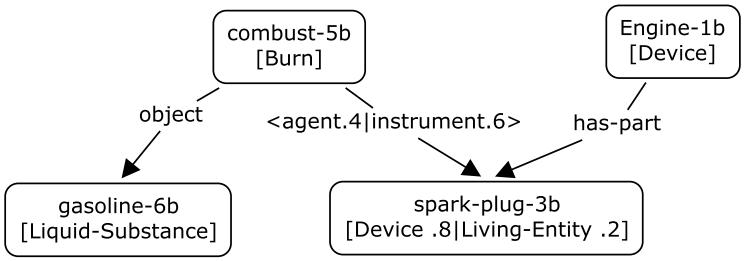


Figure 4.5: Packed representation for S2, *"The engine's spark plug combusts gasoline."*

4.1.2 Combining Packed Representations

Maintaining ambiguity within the packed representation allows us to delay commitment to an interpretation until enough evidence accrues to disambiguate. For any text fragment that results in a packed representation (PRa) containing ambiguity, there may exist other text fragments somewhere that are partly redundant, but result in a less ambiguous (or differently ambiguous) representation (PRb). The less ambiguous representation (PRb) can be used to adjust confidences in the ambiguous representation (PRa). Enough such evidence would allow us to prune unlikely interpretations, ultimately disambiguating the original representation.

Algorithm 2 describes our knowledge integration which combines two packed representations to help resolve their ambiguities. The algorithm attempts to identify their isomorphic subgraphs (redundant portions of the interpretations) and uses the information to further disambiguate their ambiguities. For illustration, we will step through Algorithm 2, merging PR1 (Figure 4.4) with PR2 (Figure 4.5).

Algorithm 2 Knowledge integration: disambiguating the packed representations

Input : PR1, PR2

Output: new packed representation

1. **Identify semantically aligned parts between PR1 and PR2.** The algorithm uses a graph matcher to identify mappings (redundant portions) between PR1 and PR2. It first identifies node mappings by aligning nodes whose base word is same or types are taxonomically aligned. From the node mappings, the same types are aligned as type mappings. It also produces the relation mappings if the relation is same and their head and tail nodes are already mapped.

2. **Use the mappings to further disambiguate PR1 and PR2.** With the currently available information (the confidence scores and the constraints in PR1 and PR2 and the mappings between them), the algorithm uses off-the-shelf joint-inference software to calculate the confidence score of each candidate interpretation. If the confidence score of one interpretation becomes much higher than the other competing ones, the interpretation is chosen while the others are discarded.

3. **Combine the disambiguated PR1 and PR2 into one packed representation using the mappings identified in the first step.**

1. The graph matcher identifies the mappings between PR1 and PR2. Some type mappings are (engine-2a[Device], Engine-1b[Device]), (spark-plug-8a[Living-Entity], spark-plug-3b[Living-Entity]), etc., and the relation mappings are ((combust-5b instrument spark-plug-3b), (ignite-3 instrument spark-plug-8)), ((ignite-3a instrument spark-plug-8a) (combust-5b instrument spark-plug-3b)), etc.

2. In this example, we use a simple joint-inference method (which is also used in our prototype system). When interpretation A is mapped with interpretation B, their confidence scores are simply added and the resulting score is assigned to both [2]. For example, the final score of DEVICE in `engine-2a` becomes 1.7 because its original score (.7) is added to the score of DEVICE in Engine-1b (1), while the confidence score of LIVING-ENTITY in `engine-2a` is unchanged. Because the score of DEVICE (1.7) is much higher than the score of LIVING-ENTITY (.3) [3], DEVICE is confirmed while LIVING-ENTITY is discarded. Confirming/pruning an interpretation may affect the other interpretations due to the constraints. Thus, deleting LIVING-ENTITY causes deletion of the *agent* relation between `ignite-3a` and `engine-2a` due to the dependency constraint A $\rightarrow$ C.

[2]Note that, in our prototype system, the scores of the type candidates are the word count generated by the Lesk algorithm (not a probability). In Section 4.2.3, we present more sophisticated combination method based on Markov Logic Network [124]

[3]In our prototype, we set the pruning threshold at $\frac{1}{3}\times$the score of the top-scored interpretation.

3. The disambiguated PR1 and PR2 are merged into a single packed representation (PR3) based on the previous mappings. Any remaining ambiguity could simply be left in PR3, possibly to be resolved with another sentence.

4.1.3 Ally: Prototype System

To evaluate our approach, we built a prototype system, Ally, by extending Kleo. Ally differs from Kleo primarily in that Ally uses the packed representation and Algorithm 2 to manage multiple candidate interpretations based on redundancy.

4.1.3.1 Parser

Ally uses the Stanford Parser [79]. To capture structural ambiguity for our experiments, we manually converted the parser output to a syntactic packed representation by adding corrections as alternatives wherever the parse tree was incorrect. This gave a syntactic packed representation with both incorrect and correct alternatives. We arbitrarily gave the original, incorrect alternatives high confidence scores and the added, correct alternatives low scores. This approach simulates the situation in which the parser pruned the correct interpretation in favor of an incorrect one with a higher confidence score. The syntactic packed representation for S1 is shown in Figure 4.6. Later in Section 4.2.3, we present an algorithm that produces the syntactic packed representation in a fully automated way.

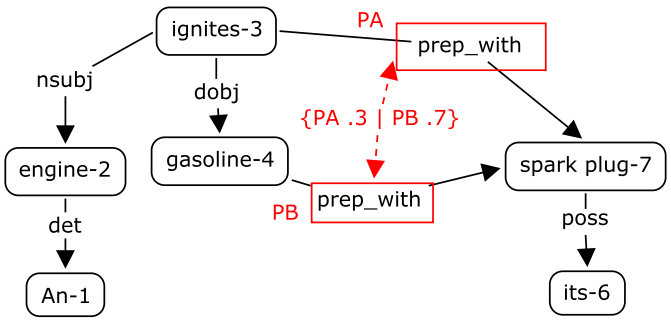


Figure 4.6: Syntactic packed representation for S1, capturing the prepositional phrase attachment ambiguity of *"with its spark plug"*.

4.1.3.2 Semantic interpreter

The semantic interpreter assigns types to nodes in the syntactic packed representation and semantic relations to the edges.

- **Type ambiguity.** Types and confidence scores are assigned to words using SenseRelate [112], off-the-shelf Lesk-based WSD software. Assigned senses are then mapped to the Component Library ontology using its built-in WordNet mappings.

- **Relational ambiguity.** Semantic relations are assigned to the dependency relations in the syntactic packed representation according to hand-written semantic interpretation rules (see Section 3.1.1). Most of the rules consider the types of the head and the tail as well as the dependency relation, but do not produce confidence scores. Ally simply scores candidates equally. A more sophisticated scoring method such as the one in Section 4.2.3 can be plugged in easily.

- **Structural ambiguity.** Parse ambiguities (such as PA vs. PB in Figure 4.6) are converted directly to structural ambiguity representations (D vs. E in Figure 4.4) in the semantic packed representation.

- **Simple Dependency.** A dependency is installed between a type t for word w and a semantic relation r when (1) r is produced by a rule based on t and (2) r is dependent on no other candidate type for w. In Figure 4.4, a dependency relation is installed from A to C, because (1) LIVING-ENTITY in `engine-2a` was used in the rule assigning *agent* between `ignite-3a` and `engine-2a` and (2) the assignment of *agent* is not dependent on DEVICE, the other candidate type of `engine-2a`.

- **Mutual dependency.** If multiple interpretations depend on one another, a mutual dependency set is created to include them.

4.1.3.3 Knowledge integration

This module implements Algorithm 2 to combine packed representations from multiple sentences. The packed representation for each sentence is merged with the combined packed representation from previous sentences. The global packed representation integrates sentence-level packed representations to the extent that they align semantically. In the worst case (completely unrelated sentences), the global packed representation would simply be the union of individual packed representations. The extent to which the global packed representation is more coherent reflects redundancy and semantic overlap in the sentences.

4.1.4 Experiment 1

The packed representation allows the system to delay ambiguity resolution. Redundancy and semantic overlap in subsequent sentences should allow Algorithm 2 to adjust the confidence in ambiguous alternatives given more context.

We first wanted to evaluate our hypothesis that Algorithm 2 can improve the interpretation accuracy when the system is given a handful of redundant texts. To do this, we manually generated a set of ten redundant texts [4] by having volunteers rewrite a short, tutorial text, using Amazon Turk (`http://mturk.com`) [5]. The volunteers had no knowledge of the purpose of the task, and were asked simply to rewrite the text using "different" language. The original text and one volunteer's rewrite is shown in the following:

> **Original Text** Hearts pump blood through the body. Blood carries oxygen to organs throughout the body. Blood leaves the heart, then goes to the lungs where it is oxygenated. The oxygen given to the blood by the lungs is then burned by organs throughout the body. Eventually the blood returns to the heart, depleted of oxygen.
>
> **Paraphrase** The heart begins to pump blood into the body. The blood first travels to the lungs, where it picks up oxygen. The blood

[4] The ten texts are presented in Appendix A.

[5] In practice, we envision a system whose task is to develop a model of a particular topic by interpreting multiple tutorial texts on the topic. Such a system might be given a clustered set of documents on the topic. Alternatively, given a single tutorial text on a topic, a system could perform its own information retrieval to collect a small corpus of texts with some confidence in their semantic overlap.

> will then be deposited into the organs, which burn the oxygen. The blood will then return to the heart, where it will be lacking oxygen, and start over again.

The total number of sentences over the ten texts was 37. Average sentence length was 14.5 words.

Evaluation Procedure

We ran two systems over the ten texts. The baseline system commits to the highest scoring consistent interpretation after each sentence. Ally produces an ambiguity-preserving packed representation. As Ally reads each sentence, it uses Algorithm 2 to merge the packed representation of the sentence with that of the previous sentences. After N sentences (varying N from 1 to 37), Ally is forced to commit to the highest scoring consistent interpretation from the packed representation. For N=1 (Ally is forced to commit after reading the first sentence), both the baseline and Ally produce the same result. For N=2, the baseline system produces the union of the highest scoring interpretations for each of the first two sentences in isolation. Ally produces a merged packed representations for the first two sentences and then prunes to the highest scoring alternatives.

At each value of N, we measured the correctness of the interpretations (the percentage of correct semantic triples) committed to by each system by comparing the committed triples against human-generated gold standard triples for the texts.

	baseline	Ally
nodes with the correct type	76 %	**91 %**
edges with the correct relation	74 %	**88 %**

Table 4.1: Percentage of nodes and edges maintaining the correct types and semantic relations in the baseline system and Ally for all 37 sentences.

We repeated the experiment ten times with different random orderings of the 37 sentences, averaging the results.

Evaluation Result

Figure 4.7 shows that the quality of both type assignment and semantic relation assignment by Ally increases as the system acquires more evidence from other sentences. This result confirms our hypothesis that delaying commitment to an interpretation resolves ambiguities better by avoiding overly aggressive pruning.

To determine an upper bound of correctness for Ally, we inspected the packed representations to see how many alternative sets within the packed representations still contained the correct interpretation, even if it is not the highest scoring alternative. This number is different from the correctness score in Figure 4.7, which is the percentage of gold standard triples in the packed representation after committing (pruning) to the highest scoring alternatives.

Table 4.1 shows that 91% of the nodes in the packed representation contain the correct type (though not necessarily the highest scoring). 88% of the edges contain the correct semantic relations among the alternatives. In contrast, the baseline system has pruned away 24% of the correct types and

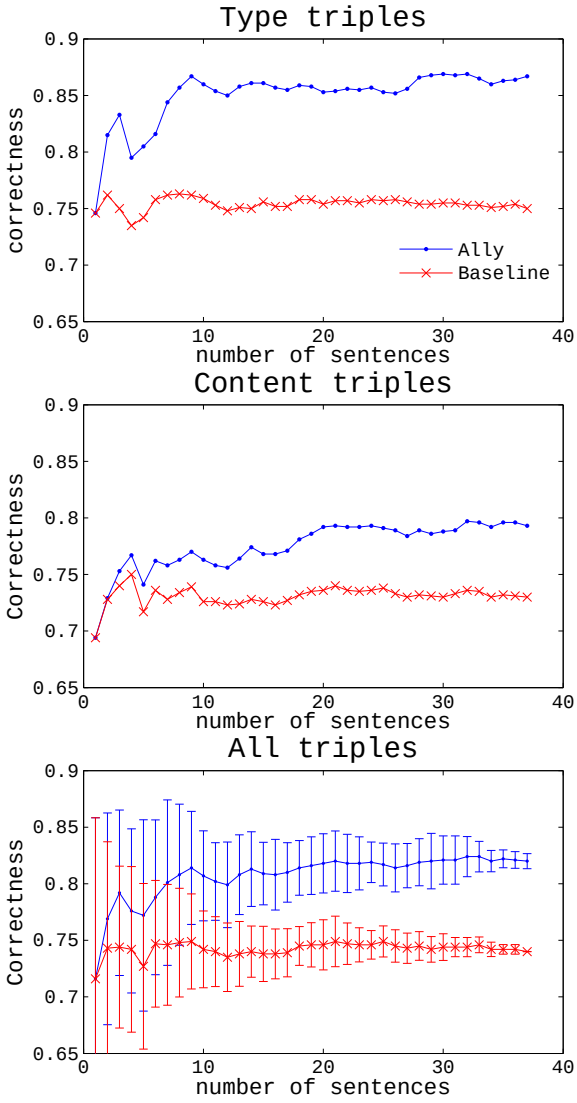


Figure 4.7: Correctness scores for Ally vs. baseline system on (a) type triples (type assignment task), (b) content triples (semantic relations assignment task) and (c) all triples (with the bars representing standard deviation). X-axis represents the values of N (from 1 to 37) and Y-axis represents the ratio of the correct triples.

26% of the correct semantic relations.

4.1.5 Experiment 2

Our second experiment aims to evaluate the claim that Ally can efficiently manage a large number of alternative interpretations. The top line in Figure 4.8 shows the number of triples in the packed representations input to Ally. This is the total number of triples (including ambiguity alternatives) in the packed representation for each sentence prior to invoking Algorithm 2. The middle line is the number of triples remaining after merging and pruning by Algorithm 2. The bottom line is the number of triples after pruning all but the highest scoring alternatives (the baseline system). The results show that Algorithm 2 achieves significant compression over unmerged packed representations. The resulting size of the merged packed representations more closely tracks the size of the aggressively pruned representations.

4.1.6 Experiment 3

Finally, we wanted to measure the sensitivity of our approach to the quality of natural language interpretation. In this experiment, we artificially varied the confidence scores for the correct interpretations in the packed representations input to Ally and the baseline system by a fixed percentage. For example, consider a node `heart-1` in a packed representation. Among the candidate types is the correct sense for its context: INTERNAL-ORGAN with confidence 0.8. We reran Experiment 1 varying the confidence in INTERNAL-

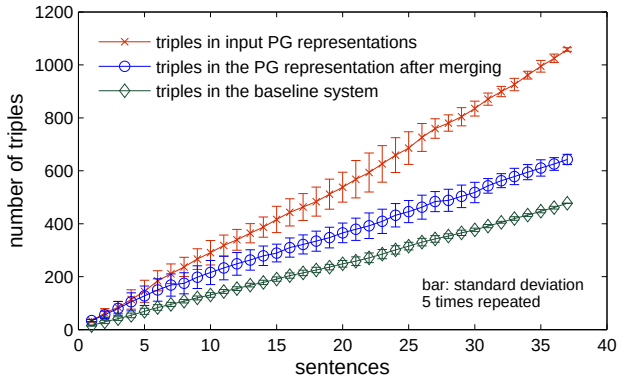


Figure 4.8: Total number of triples in individual sentence packed representations (top); total number of triples in the packed representation after merging in Ally (middle); total number of triples after pruning to the highest scoring alternative (bottom).

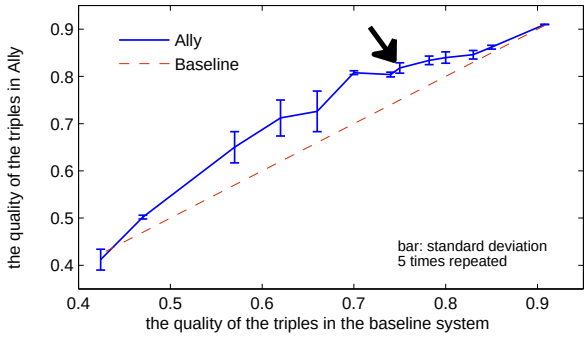


Figure 4.9: Sensitivity of Ally and the baseline system to the quality of the NL system output. The quality of the triples produced by the NL component is perturbed, affecting performance accuracy of the two systems. For example, when the quality of the NL output is perturbed to the level of 70% accuracy, Ally achieves a higher accuracy, 80%, using Algorithm 2. The arrow indicates unperturbed language interpreter performance.

ORGAN in increments of both +10% and -10%, while scaling the confidences in the incorrect types equally. As the confidence in correct interpretations is increased, all correct interpretations become the highest scoring, so aggressive pruning is justified and the baseline system performance approaches Ally's performance. As the confidence in correct interpretations is decreased, these interpretations are more likely to be pruned by both systems.

Figure 4.9 shows that Algorithm 2 is able to recover at least some correct interpretations even when their original scores (relative to incorrect alternatives) is quite low.

4.1.7 Summary

We presented the packed representation to delay ambiguity resolution beyond sentence and text boundaries. Improvements in the correctness of semantic interpretation of sentences is possible without an explosion in size when maintaining multiple interpretations. Finally, our experiment shows that redundancy is useful information for resolving ambiguities in packed representation.

4.2 Using OntoNotes

A semantically annotated corpus annotates texts with semantic information such as word senses [69], semantic roles [111], and temporal/causal relations [17]. These annotations could be useful for resolving the ambiguities in the packed representation by providing statistics about semantic interpreta-

tions for certain contexts. Moreover, the quality of the annotations is generally good because they are built by linguistic experts manually.

For our study, we chose a corpus, OntoNotes [69], and evaluated its semantic annotations – the word sense and semantic role annotations [6]. We chose OntoNotes for three reasons. First, its annotation quality is good, achieving 90% inter-annotator agreement. Second, it contains several types of semantic annotations, thereby providing statistical information about relationship among different semantic annotations. Finally, it is large enough to provide sufficient data for an empirical study.

To evaluate the benefits of using OntoNotes, we built a prototype system, Ally [7]. Ally maintains multiple candidate interpretations using packed representation, avoiding premature pruning. At the end of the pipeline, Ally jointly resolves the remaining ambiguities with the word sense and semantic relation annotations in OntoNotes to identify the best overall semantic representation. This approach can improve the accuracy of semantic interpretation by exploiting more information downstream, rather than imprudently resolving ambiguities upstream. Note that semantic resources, such as OntoNotes, can almost never be used in upstream components (such as the parser), because the resources contain no syntactic information.

In our experiment, we compared Ally with a baseline system that also

[6]Because we wanted to use only a semantic knowledge resource, we did not use the parsing annotations. For the remainder of this dissertation, when we refer to OntoNotes, we are referring only to its word sense and semantic relation annotations.

[7]For convenience, we use the same name as the system introduced in Section 4.1.

uses OntoNotes but is based on the traditional pipeline approach, committing to interpretations after each step of the pipeline. Our experiment shows that Ally produces more accurate semantic representations than the baseline system because it delays ambiguity resolution using packed representation and resolves the ambiguities jointly with OntoNotes.

In the following sections, we explain OntoNotes (Section 4.2.1) and then present another packed representation scheme (Section 4.2.2). Then, we present our prototype system, Ally, which uses OntoNotes to disambiguate the packed representation (Section 4.2.3). Finally, we present the positive experimental results to show that Ally outperforms the traditional pipeline approach (Section 4.2.4 through 4.2.4).

4.2.1 OntoNotes

OntoNotes is a large-scale, multi-lingual corpus annotated with a variety of information: parses, word senses, semantic roles, and co-references. In our study, we use only word sense and semantic role annotations. One advantage of OntoNotes is its high annotation quality – 90% inter-annotator agreement. Three versions have been published, and we have used the second version, OntoNotes 2.0 [8] in our study. OntoNotes 2.0 comprises various genres of text such as news, conversational telephone, webblogs, use net, broadcast, talk shows from 300K of the Penn Treebank 2 Wall Street Journal corpus, and 200K from the TDT4 corpus.

[8] In this dissertation, OntoNotes refers to OntoNotes 2.0.

Sentence : The plant and another next door changed the face of Postvilles. (from abc0001)

// word sense annotation

```
1 plant-n 1
6 change-v 1
8 face-n 5
```

// semantic role annotation

```
6 gold change-v change.01 ----- 0:2-ARG0 6:0-rel 7:2-ARG1
```

Figure 4.10: Example of the word sense and semantic role annotations in OntoNotes. The sense annotation, `1 plant-n 1`, represents that *plant* (index: 1) is a noun (plant-n) and that its sense choice is the first sense. The semantic role annotation (last line) represents that *change* (index: 6) is a verb (change-v), maps to the frame, `change.01`, and has two roles: ARG0 (*"The plant and another next door"*) and ARG1(*"the face of Postvilles"*). 0:2 represents the word span covered by a non-terminal in the constituency parse tree – the grandfather (indicated by 2) of the first word (indicated by 0).

The word sense inventory in OntoNotes contains sense information for approximately 1,474 words, which are nouns and verbs frequently used. Sense distinction in OntoNotes is more coarse-grained than WordNet [103] to improve the inter-annotator agreement. For semantic role anntation, OntoNotes uses 3,656 frames and their semantic roles from Propbank [111]. Figure 4.10 shows an example of the word sense and semantic role annotation. For more information about OntoNotes, see [69].

4.2.2 Packed Representation

Our second packed representation differs from the previous version (Figure 4.4) primarily in that it aims to represent the derivation processes

compactly – the derivation trees in (b) in Figure 4.1 – whereas the previous version compresses the semantic representations resulting from the derivation processes. Because of this difference, the new version can represent the constraints in more fine-grained ways, but may be less compact. One advantage of this representation is that it incorporates the representations of the parsing ambiguities automatically produced (whereas the previous version relies on manual encoding to represent them).

Figure 4.11 shows an example of this packed representation for the interpretation of the sentence, *S1: "The man saw the boy with his glasses"*. It consists of two parts, the base representation and the constraints. The base representation is a graphical representation with two types of semantic information – word senses (in the nodes) and semantic relations between pairs of words (in the edges). A variable is introduced to represent an ambiguous interpretation. For example, the variable T1 represents that the system has not yet committed to a specific word sense for *glasses*.

As in the previous version, there are two types of constraints, ambiguity constraints and relationship constraints. The ambiguity constraints enumerate the possible candidates for each variable, and the relationship constraints describe the relationship among the candidates. We first explain the ambiguity constraints.

1. **Parsing ambiguity:** This constraint describes parsing ambiguities. For example, (PARSEXOR with (p1 .6) (p2 .4)) describes the ambiguity of

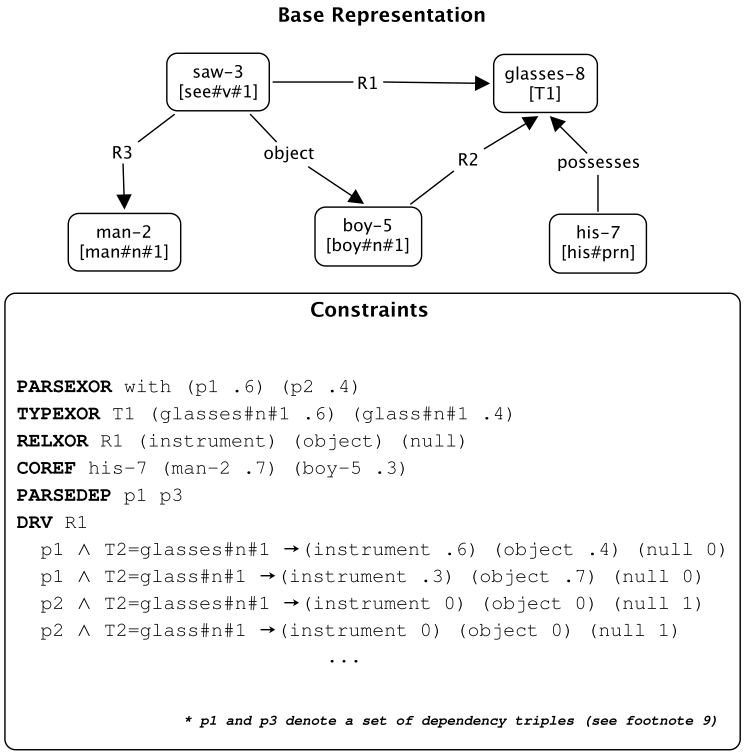


Figure 4.11: A packed representation

attaching the *with* prepositional phrase. p1 and p2 denotes the attachment to *saw* (with score .6) and *boy* (with score .4) respectively [9]

2. **Word sense ambiguity:** This constraint describes ambiguities when assigning a word sense. For example, (TYPEXOR T1 (glasses#n#1 [10] .6) (glass#n#1 .4)) represents that the word sense for *glasses* can be glasses#n#1 (with the score .6) or glass#n#1(with the score .4), but not both. Unlike the previous version, we use the word senses rather than the semantic types.

3. **Semantic relation ambiguity:** This constraint describes ambiguities in assigning a semantic relation. For example, (RELXOR R1 (instrument) (object) (null)) represents that three candidates are maintained for the semantic relation between *saw* and *glasses*: instrument, object and null (no relation). This example does not include numerical scores but instead uses a DRV constraint to describe how the scores can be calculated from the other constraints.

4. **Co-reference ambiguity:** This constraint describes ambiguities when assigning a co-reference link. For example, (COREF his-7 (man-2 .7) (boy-5 .3)) represents that *his* may refer to *man* (with score .7) or *boy* (with score .3) .

[9] p_1 and p_2 (called PIDs, as explained in Section 4.2.3.1) indicate a set of dependency triples, {(saw-3 prep_with glasses-8)} and {(boy-5 prep_with glasses-8)}, respectively.
[10] <lemma>#<part-of-speech>#<sense number>

The relationship constraints describe the relationship between different ambiguities.

5. **Dependency in parsing:** This constraint describes the dependency between two parse fragments. For example, (PARSEDEP p1 p3) represents that the triples denoted by p1 depend on the ones denoted by p3. Therefore, if any dependency triple in p3 turns out to be false, the triples in p1 should be discarded, too.

6. **Derivation:** This constraint describes how the candidates and their scores are derived from the interpretations made upstream. For example, the DRV constraint in Figure 4.11 represents that the candidates and their scores for R1 are derived based on the decision between p1 and p2 (the first PARSEXOR constraint) and the assignment in T1. Each row describes the scores of the candidate relations when the upstream components make a different choice. For example, the first row indicates that the scores of *instrument*, *agent* and *null* in R1 are .6, .4 and 0 respectively when p1 is chosen and T1 is glasses#n#1.

4.2.3 Ally: A Prototype Language Interpreter

Ally interprets English texts using standard components arranged in a conventional pipeline (Figure 4.12). However, unlike traditional pipeline systems, the components in Ally produce packed representations of a set of possible interpretations, instead of committing to a single one.

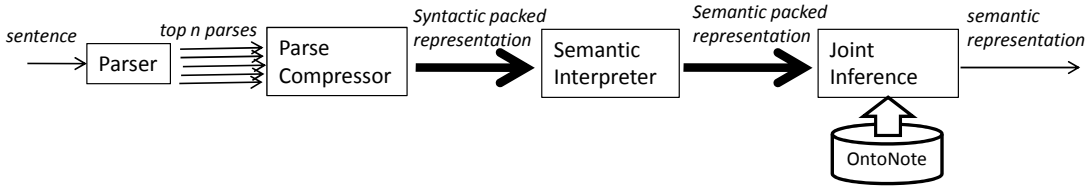


Figure 4.12: The architecture of Ally

Given an English sentence, Ally first uses the Stanford Parser [79] to produce the top n parses. Ally compresses the parses into a single representation called the syntactic packed representation. Then, Ally's semantic interpreter infers semantic information, word senses and semantic relations, to produce the semantic packed representation. Ally maintains multiple candidates for ambiguous word senses and semantic relations. Finally, Ally resolves ambiguities jointly to select the overall best interpretation. This joint resolution step also uses additional knowledge from OntoNotes (the word sense and semantic relation annotations). The final output is a semantic representation in which word senses and semantic relations are grounded in the sense and relation inventory of OntoNotes.

4.2.3.1 Producing syntactic packed representation

For each sentence in a text, Ally uses the Stanford Parser to produce the top 40 parses (P1, P2, ..., P40) and then packs them into a syntactic packed representation (see Figure 4.14). This representation has the same structure as the semantic packed representation (see Figure 4.11), except that it uses dependency relations and only two constraints, PARSEXOR and PARSEDEP.

P1	P2	P3
(saw-3 nsubj man-2)	(saw-3 nsubj man-2)	(saw-3 nsubj man-2)
(saw-3 dobj boy-5)	(saw-3 dobj boy-5)	(saw-3 dobj boy-5)
(saw-3 prep_with glasses-8)	**(boy-5 prep_with glasses-8)**	**(saw-3 dep glasses-8)**
(glasses-8 POS NN)	**(glasses-8 POS NN)**	**(glasses-8 POS NNP)**

Figure 4.13: Three candidate parses. To save space, some triples common to all three parses are omitted. The bold font indicates the differences among the parses.

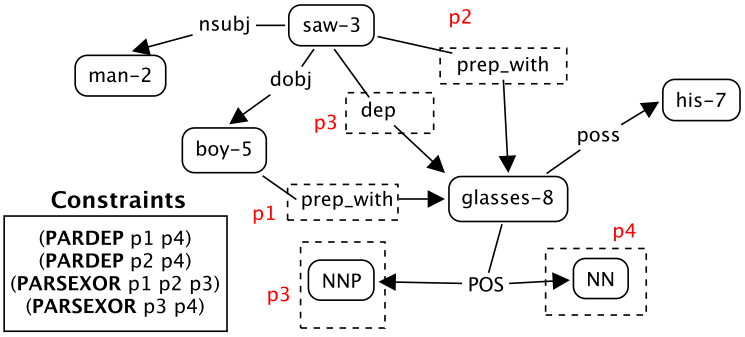


Figure 4.14: Syntactic packed representation for S1

We describe the packing algorithm step-by-step, showing how the three candidate parses shown in Figure 4.13 are packed to produce the syntactic packed representation shown in Figure 4.14. The packing algorithm can be applied to any dependency parser that generates multiple candidate parses.

Step 1: Tag the triples with parse IDs.

Tag each triple with the IDs of the parses in which it occurs, as shown in Figure 4.15. For example, (glasses-8 POS NN) is tagged with {P1, P2} because it appears in two candidate parses, P1 and P2. We will call this tag a PID.

(saw-3 nsubj man-2)	{P1, P2, P3}
(saw-3 dobj boy-5)	{P1, P2, P3}
(saw-3 prep_with glasses-8)	{P1}
(glasses-8 POS NN)	{P1, P2}
(boy-5 prep_with glasses-8)	{P2}
(saw-5 dep glasses-8)	{P3}
(glasses-8 POS NNP)	{P3}

Figure 4.15: The result of step1

{P1, P2, P3}	::=	(saw-3 nsubj man-2)
		(saw-3 dobj boy-5)
{P1, P2}	::=	(glasses-8 POS NN)
{P1}	::=	(saw-3 prep_with glasses-8)
{P2}	::=	(boy-5 prep_with glasses-8)
{P3}	::=	(saw-3 dep glasses-8)
		(glasses-8 POS NNP)

Figure 4.16: The result of step2

Step 2: Group together the triples with the same PID.

Figure 4.16 shows the result of this step.

The next two steps generate the PARSEDEP and PARSEXOR constraints.

Step 3.1: Produce PARSEDEP constraints.

For every pair of PIDs, A and B, generate the constraint (PARSEDEP A B) if A is a proper subset of B. For example, Ally generates (PARSEDEP {P1} {P1, P2}) because {P1} is a proper subset of {P1, P2}. The PID that corresponds to the universe set ({P1, P2, P3}, in our example) is ignored.

The purpose of this step is to create the PARSEDEP constraints (de-

pendency between two parse fragments) that may later be used to prune candidate parses. (PARSEDEP {P1} {P1, P2}), for example, represents that the existence of the triples defined by {P1} (i.e. (saw-3 prep_with glasses-8)) depends on the validity of the triples defined by {P1, P2} (i.e. (glasses-8 POS NN)). Therefore, if Ally later finds that the part-of-speech of glasses-8 is not NN (i.e., P1 and P2 are incorrect), it can also discard (saw-3 prep_with glasses-8). The following shows the result of this step, when applied to the PIDs in Figure 4.16.

(PARSEDEP {P1} {P1, P2})
(PARSEDEP {P2} {P1, P2})

Step 3.2: Produce PARSEXOR constraints.

For every pair of PIDs, A and B, generate the constraint (DISJOINT A B) if A and B are disjoint. (DISJOINT A_1 A_2 ... A_n) means that if A_i is found to be true, the others should be false. Notice a difference between DISJOINT and PARSEXOR. PARSEXOR further asserts the converse of the implication: if and only if A_i is found to be true, the others should be false. When applied to the PIDs in Figure 4.16, this step produces:

(DISJOINT {P1} {P2})
(DISJOINT {P2} {P3})
(DISJOINT {P1} {P3})
(DISJOINT {P1, P2} {P3})

The algorithm then combines DISJOINT constraints to produce a PARSEXOR constraint. First, it identifies a maximal DISJOINT set, in which

all the PIDs are in DISJOINT relationships to one another. For example, (DISJOINT {P1},{P2}, {P3}) is created because {P1},{P2}, {P3} are in a DISJOINT relation with one another.

(DISJOINT {P1},{P2}, {P3})
(DISJOINT {P1, P2},{P3})

Then, the algorithm converts each DISJOINT constraint into a PARSEXOR constraint in the following way. If the union of the PIDs in the DISJOINT constraint equals the universe set, the predicate name is simply changed from DISJOINT to PARSEXOR. This is because the union (the universe set) is assumed to contain the correct parse and therefore a PID should be true if the others are found to be incorrect.

If the union of the PIDs in the DISJOINT constraint is a proper subset of the universe set, the correct parse may be out of the union, and therefore a PID cannot be confirmed, even though the others are found to be incorrect. To represent it, the algorithm changes the predicate name from DISJOINT to PARSEXOR and also inserts ϕ as one of its arguments to denote that a PID may not be true even though all the others are false. The result of this step is

PARSEXOR({P1},{P2}, {P3})
PARSEXOR({P1, P2},{P3})

Step 4: Replacing PID with a unique id.

Figure 4.14 shows the result of this step. It shows four PIDs – p1, p2, p3 and p4 – that correspond to {P1}, {P2}, {P3}, {P1, P2} respectively.

4.2.3.2 Assigning word senses and semantic relations

In this step, Ally produces a semantic packed representation from a syntactic one. For each word (in the nodes of the syntactic packed representation), Ally ranks the candidate senses using SenseRelate [112]. If a candidate's score is much lower than the top score (less than 1/3 of the top score), Ally discards that sense. If multiple candidate senses are still left, Ally creates a TYPEXOR constraint. Because SenseRelate outputs WordNet [103] senses, Ally converts them into OntoNotes-based senses using the mappings provided by OntoNotes.

To convert a dependency relation into an OntoNotes semantic relation, we built a semantic-relation labeler by training a Bayesian network [19] on the first 30% of the documents in each section of OntoNotes. We used the following features: lemma, part-of-speech, word senses of the head node and the tail node, and the dependency label. The training accuracy in ten fold cross-validation was 75%.

Because the resulting semantic relation depends on the senses of the head node and tail node and the existence of the dependency triple, the labeler produces the candidate semantic relations (along with their scores) for all possible combinations and then creates a DRV constraint.

We did not produce co-reference links.

4.2.3.3 Knowledge integration

After producing the semantic packed representation, Ally resolves the ambiguities in the representation with the sense and semantic relation annotations of OntoNotes and a joint-inference method. The system first uses heuristics to resolve some of the ambiguities with OntoNotes, if they can be reliably resolved. The remaining ambiguities are then jointly resolved based on the decisions made by OntoNotes and the constraints in the packed representation.

Heuristics for using OntoNotes

Ally applies simple heuristics to evaluate each candidate interpretation using OntoNotes [11]. The heuristics measure the frequency of the candidate interpretation in OntoNotes for a similar context. For a candidate word sense S of the word in a node, Ally first extracted from the packed representation the semantic triples containing S. A semantic triple is (word sense in the head node, semantic relation, word sense in the tail node). For example, for the glasses#n#1 candidate in T1, the extracted semantic triples are (saw[see#v#1] instrument glasses[glasses#n#1]), (saw[see#v#1] object glasses[glasses#n#1], ..., (his[his#prn] possesses glasses[glasses#n#1]). Then, Ally simply counted their occurrences in OntoNotes. If the count for one interpretation is much higher (3x) than the count for the others, it was selected.

[11] There could be more sophisticated methods for using annotated corpus (e.g., [42]), but the simple heuristics sufficed for our purpose because our focus was on delay of ambiguity resolution not on efficient use of the corpus.

Otherwise, the decision was delayed until the joint inference step (which will be explained next).

Similarly, for a candidate semantic relation R in an edge, Ally extracts the semantic triples containing R from the packed representation and then counts their occurrences in OntoNotes. If one candidate relation's count is much higher than the others, it is selected. Otherwise, the decision is made by the joint inference step.

Joint inference

After OntoNotes resolves some ambiguities, these new decisions are used as evidence for the system to resolve the remaining ambiguities. In this step, the system jointly resolves the remaining ambiguities through the constraints of the packed representation.

For this task, we use Alchemy, a probabilistic inference engine [12] based on Markov Logic Networks [124]. Ally translates the constraints of the packed representation into Alchemy statements. Then, it performs Maximum A Posterior (MAP) inference to choose the interpretations that maximize the overall posteriori probability, given the interpretations selected by OntoNotes as observed evidence. Appendix B shows how the constraints are translated into the Alchemy statements.

[12] `http://alchemy.cs.washington.edu`

4.2.4 Experiment 1

This evaluation measures the contribution of using the packed representation to use OntoNotes. We compare the performance of Ally against two baseline systems that commit to an interpretation at each step of the pipeline. The performance metric is accuracy of word sense disambiguation and semantic relation assignment.

Baseline systems

Baseline1 uses the same components as Ally but commits to the highest-scored interpretation at each step. It does not use OntoNotes or the joint inference method.

Baseline2 is same as Baseline1 except that it uses OntoNotes in each component. For fair comparison, Baseline2 uses OntoNotes in the same manner as Ally. The following list describes how each component in Baseline2 benefits from OntoNotes.

- **Parsing** OntoNotes is not used in this step because there is no straightforward way to correct a wrong parse using semantic annotations. One advantage of delaying ambiguity resolution is that Ally, unlike Baseline2, can use OntoNotes later in the pipeline to select among the alternate parses that are maintained in the packed representation.

- **Word sense disambiguation** For a candidate sense S of a word, Baseline2 extracts the dependency triples that contain the word. Then, Base-

line2 counts the triples in OntoNotes that can be aligned with any of the dependency triples. To be aligned, the head and the tail lemmas must be the same and the sense of either the head or the tail word must be S. As with Ally, if the count for one candidate interpretation is much higher than the others, it is chosen. Otherwise, the choice by SenseRelate remains unchanged.

- **Semantic relation assignment** For a candidate relation R, Baseline2 counts the occurrences of the triple, <the sense in the head node, R, the sense in the tail node>, in OntoNotes. If the count for one candidate relation is much higher than the others, it is chosen. Otherwise, the choice by the semantic relation labeler remains unchanged.

Experimental setup

The test documents were texts $33 \sim 60$ in the ABC section of OntoNotes, excluding a few sentences that failed to parse. They contain 405 sentences with an average length 16.9 words. The annotations in the remaining documents (except the ones used in the training of the semantic relation labeler) were used as the knowledge resource in the joint inference step. The three systems each produced semantic representations for the sentences in the test documents.

We measured the accuracy of word sense disambiguation (WSD) and semantic relation assignment (SR) by comparing the semantic representations

	Baseline1	Baseline2	Ally
WSD	.37 (304/787)	.46 (364/787)	**.51 (402/787)**
SR	.41 (568/1398)	.44 (615/1398)	**.48 (674/1398)**

Table 4.2: Evaluation Result

with the OntoNotes gold standard annotations:

$$\text{WSD accuracy} = \frac{\text{total number of word senses correctly identified}}{\text{total number of word senses tested}}$$

$$\text{SR accuracy} = \frac{\text{total number of relations correctly identified}}{\text{total number of relations tested}}$$

We tested 787 word senses and 1398 semantic relations.

Experiment result

Ally outperformed the two baseline systems in both WSD and SR. See Table 4.2 [13].

Two factors contributed to Ally's success. First, because Baseline2 outperformed Baseline1, we conclude that the use of OntoNotes to select among competing interpretations was useful, especially for WSD. Second, because Ally outperformed Baseline2, we conclude that the use of the packed representation to delay ambiguity resolution was useful, both for WSD and SR.

[13]State-of-the-art methods for using OntoNotes achieve the accuracy of 89.1% for word-sense disambiguation [155] and the F-score of 86.02% for semantic role labeling [30]. These method use more sophisticated machine learning models (support vector machine, maximum entropy classifier) with more linguistic features beyond simply counting frequencies. It is our future research to investigate whether the improvement in the interpretation accuracy can also be achieved when these state-of-the-art methods are used instead of our simple heuristics.

	Packed Representation
WSD	.82 (645/787)
SR	.71 (996/1398)

Table 4.3: The upper bounds on Ally's performance. The ratio of the nodes containing the correct type (WSD) and the edges containing the correct relation (SR) in the packed representation

There is considerable opportunity to further improve WSD and SR in Ally. We measured the upper-bounds on Ally's accuracy for these tasks by counting the number of times that the correct interpretation is in the packed representation, even if Ally does not select it. See Table 4.3. Ally's use of the packed representation almost doubles the chance of preserving the correct interpretation, as compared with Baseline2 (the second column in Table 4.2). Further research on more sophisticated methods of using OntoNotes, and other semantic resources, is warranted.

Finally, Ally's heuristics for using OntoNotes (as described in Section 4.2.3.3) – even though they are simple – were effective. They achieved an accuracy of 89.7% for word senses and 74.6% for semantic relations when measured for cases in which the interpretations are committed to by OntoNotes.

4.2.5 Experiment 2

We also wanted to evaluate our parse packing algorithm described in Section 4.2.3.1.

First, we measured the degree of compression achieved by the algorithm. For the first 90 sentences from the Brown Corpus [81], we generated

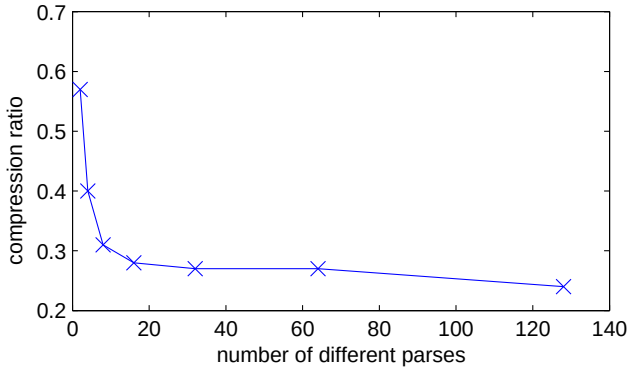


Figure 4.17: Compression ratio as the number of the alternative parses varies

the top k alternative parses using the Stanford Parser (varying k in 2, 4, 8, 16, 32, 64, and 128). Then, we calculated the compression ratio by dividing the size of a syntactic packed representation (number of dependency triples + number of PIDs + number of constraints) by the total size of the alternative parses (total number of triples in all candidate parses). Figure 4.17 shows the result of this calculation: the compression ratio nears 25% as the number of alternative parses increases.

We also evaluated the benefits of the constraints in the syntactic packed representation. One advantage of the constraints is that they can inform the system of which triples should be validated to find the correct parse, without having to check all alternative parses individually. Specifically, the ambiguity constraints (PARSEXOR) provide information about which triples are mutually exclusive, and the relationship constraints (PARSEDEP) allow the system to confirm/reject triples when one of them changes.

To measure this benefit, we calculated the number of triples the system

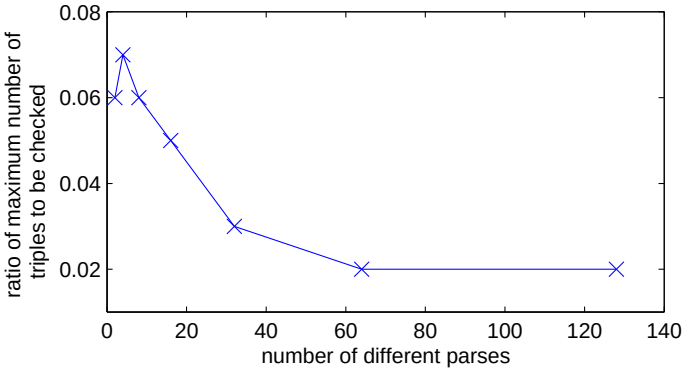


Figure 4.18: Compression ratio

should validate in order to locate the correct parse. To do this, we simply counted the number of dependency triples defined by the PIDs (which would overestimate the actual number of the triples that should be validated) and then calculated the ratio of the number of those triples to the total number of dependency triples in all candidate parses.

The result, shown in Figure 4.18, is that only 2 - 7% of the dependency triples in the candidate parses are referenced by the constraints, indicating that only a small number of dependency triples need to be validated to locate the correct parse, and that the differences among the top k alternative parses are minimal.

4.2.6 Summary

Our experiment shows the benefit of using OntoNotes for resolving the ambiguities in the packed representation. Our prototype system used only one type of evidence from one semantic resource, which suggests that using more

evidence might further improve interpretation accuracy.

4.3 Using Prismatic

External knowledge resources could be useful for ambiguity resolution. One type of knowledge resource gaining much attention recently is the one automatically constructed with information extracted from texts (e.g., taxonomic information [132], causal/temporal information [32], scripts [27], and syntactic information [48]). The quality of this type of knowledge resource may not be as good as the hand-crafted ones, but they provide broad coverage and can be built rapidly. To our knowledge, their impact on text understanding has not been evaluated thoroughly.

In this section, we present our evaluation of a knowledge resource in this category, Prismatic [48], as the source of information for resolving ambiguities in the packed representation. Prismatic is a large-scale knowledge resource that contains statistical information about syntactic relationships among the words. It also contains a shallow level of semantic information such as semantic types (e.g., PERSON, COUNTRY, DATE, etc.).

Similar to the method in Section 4.2, which uses OntoNotes for ambiguity resolution, we measure the extent to which Prismatic supports each candidate interpretation in the packed representation. The information is then incorporated when the overall best semantic representation is selected at the end of the pipeline. Our preliminary evaluation, however, shows that the improvement is not significant – our method improves the performance of type

assignment but degrades the performance of semantic relation assignment. Based on the analysis of the evaluation result, we present our future work to improve Prismatic.

This section is organized as follows. Section 4.3.1 introduces Prismatic, and Section 4.3.2 presents our approach for using Prismatic to disambiguate the packed representation. Section 4.3.3 presents our preliminary experimental results, and Section 4.3.4 presents our future work to improve Prismatic.

4.3.1 Prismatic

Prismatic is a large-scale knowledge resource automatically created from large corpora (30G), including Wikipedia, the NY Times, and Web documents. The primary units in Prismatic are Prismatic Frames, the snippets from the parse trees (see Figure 4.20 for the example frames). Prismatic also provides a shallow level of semantics by including semantic types for named entities. For example, in Figure 4.20, (b) represents that PERSON is often a syntactic subject of Receive, and (c) represents that DATE is often the object of the preposition "in" when the verb is Receive. Each Prismatic Frame is associated with a count, which indicates the frequency of the occurrences of the frame in the corpora, thereby indicating how prevalent the frame is.

We explain how Prismatic Frames are generated with the following example sentence: *"In 1921, Einstein received the Nobel Prize for his original work on the photoelectric effect"*. First,the sentence is parsed by the ESG parser [100] as shown in Figure 4.19. Then, the patterns (called frame cuts)

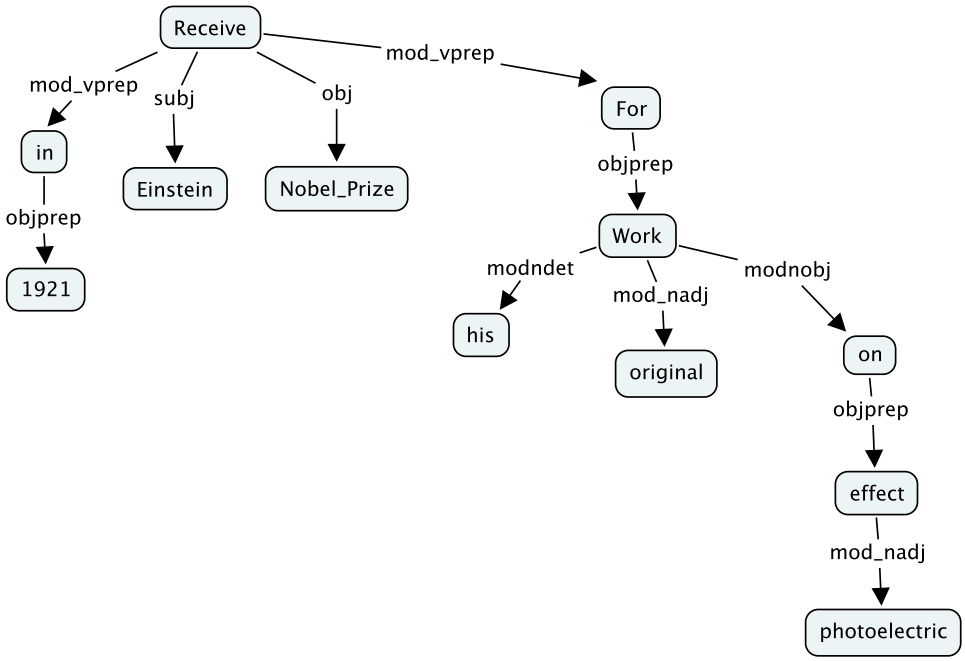


Figure 4.19: Parse tree produced by the ESG parser [100] for *"In 1921, Einstein received the Nobel Prize for his original work on the photoelectric effect"*.

are applied to the parse tree to extract Prismatic Frames. For example, (a) and (c) in Figure 4.20 show the frames produced by applying the S-V-O and S-V-P-O pattern to the parse tree in Figure 4.19, respectively (S - subject, V - verb, O - object, IO - indirect object, P - preposition). The following frame cuts are primarily used: S-V-O, S-V-O-IO, and S-V-P-O. Some Prismatic Frames are produced by replacing named entities with the semantic types assigned by a named-entity tagger. For example, (b) in Figure 4.20 is produced by replacing *Einstein* in (a) with its semantic type, PERSON.

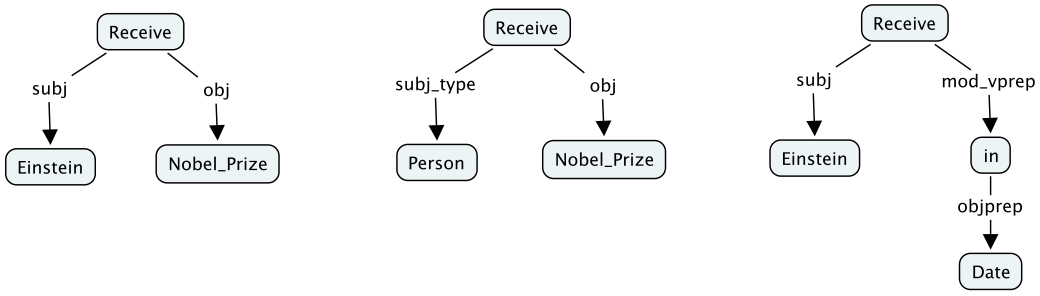


Figure 4.20: Prismatic Frames that are produced from the parse tree in Figure 4.19. (a) is produced by applying S-V-O to the parse tree. (b) is produced by replacing *Einstein* in (a) with PERSON. (c) is produced by applying S-V-P-O to the parse tree.

4.3.2 Our Approach

We use the packed representation (see Section 4.2.2) with the semantic types and semantic relations defined in the Component Library, instead of using OntoNotes.

We addressed two challenges in order to use Prismatic in the disambiguation of the packed representation. First, the scoring functions were formulated to evaluate the candidate interpretations (candidate parse fragments, candidate word senses, and candidate semantic relations) based on Prismatic. One difficulty of formulating the scoring functions is the discrepancy in the representation language between Prismatic and our packed representation. Prismatic uses the grammatical relations from the ESG Parser [100], whereas our packed representation uses the representations from two sources, the Stanford Parser's grammatical relations [96] and the Component Library's semantic types and semantic relations [11]. Sections 4.3.2.1 $\sim$ 4.3.2.3 introduce our

scoring functions, explaining how the discrepancy in the representational language is resolved.

The second challenge of using Prismatic in the disambiguation of the packed representation is jointly combining multiple scores (i.e., Prismatic scores and the scores already assigned by our text interpretation system, Ally) to extract the overall best representation from the packed representation. Section 4.3.2.4 describes our machine-learning-based approach to address this challenge.

4.3.2.1 Scoring candidate parse fragments

For each candidate dependency triple (e.g., (saw-3 prep_with glasses-8) in Figure 4.11), we measure the conditional probability of assigning the dependency relation given two words in arg1 and arg3.

$$\begin{aligned} Score([head, rel, tail]) &= P([head, rel, tail]|[head, *, tail]) \\ &= P([head, rel, tail)/P([head, *, tail]) \\ &= C([head, rel, tail])/C([head, *, tail]) \end{aligned}$$

$C(X)$ means the frequency count of the Prismatic frames containing the dependency triple, X. We resolve the discrepancy of the grammatical relations between the Stanford Parser and the ESG parser by manually mapping them.

4.3.2.2 Scoring candidate semantic types

As in most word sense disambiguation methods, we also measure the extent to which each candidate type is related to the context words (surrounding texts). Our method first acquires the words closely related to the candidate type and then measures the relatedness between those words and the context words using Prismatic. The extended list of the related words, rather than a single candidate type, is used to improve "hits" in Prismatic.

We illustrate our scoring method with an example of scoring the candidate semantic type, DEVICE, for the word *"starter"* in the sentence, *"Drew Bledsoe replaced injured starter Tom Brady"*.

Step 1: Gather the words closely related to the candidate type.

In our implementation, we use the synonyms from the WordNet synset associated with the candidate type [14]. For example, the first sense of *starter*, starter#n#1, from which DEVICE is derived, has three synonyms – *starter_motor*, *starting_motor* and *electric_motor* – and these words are used as the words related to DEVICE.

Step 2: Set the context words to be the words that are directly related to the target word in the dependency parse tree.

[14]These synsets are associated with the candidate type through the ComponentLibrary-to-WordNet mappings provided by Component Library. Section 4.1.3.2 explains this process in detail.

In our example, the context words are *injured* and *Tom_Brady*, which directly modify *starter*.

Step 3: Calculate the relatedness scores for all possible pairs between the words acquired from Step 1 and the context words. Then, average the scores.

Relatedness(w1, w2) is defined as (w1 and w2 are words)

$$weight(w1, w2) * \textit{total count of Prismatic frames containing w1 and w2}$$

$weight(w1, w2)$ is an adjusting factor to rule out the cases in which $w1$ or $w2$ may be used as a different word sense in the returned Prismatic Frames. For example, the sense of *injure* in the example sentence means *causing bodily harm* (the first WordNet sense), whereas a retrieved Prismatic Frame may contain the word used as a different sense, meaning *causing emotional damage* (the second WordNet sense). $weight(w1, w2)$ is simply defined as [15]:

$$\frac{1}{\textit{number of senses for w1 * number of senses for w2}}$$

4.3.2.3 Scoring candidate semantic relations

To bridge the gap between the syntactic (Prismatic) and the semantic relations (Component Library), we use our text interpretation system, Ally, to convert the Prismatic relations into the Component Library relations.

[15] This simple formula treats the individual word senses equally. A more accurate method may consider the frequency of the senses.

First, we gather all Prismatic frames that contain the words in arg1 and arg3. For example, for the candidate semantic relation (ignite instrument spark-plug), Prismatic may return the following Prismatic Frames: (verb=[ignite] subj=[spark-plug]), (verb=[ignite] prep=[with] prep_obj=[spark-plug]), etc.

Then, a Prismatic Frames is classified as an applicable frame if one of its syntactic relations can be converted into the candidate semantic relation. Specifically, the ESG grammatical relation in the Prismatic Frame is converted into a corresponding Stanford Parser's grammatical relation, which is, in turn, converted into a Component Library relation by the semantic interpretation rules explained in Section 3.1.1.2. The score is then calculated in the following way:

$$\frac{\textit{total counts of applicable frames}}{\textit{total counts of all frames containing arg1 and arg2}}$$

4.3.2.4 Extracting the overall best semantic representation

Our approach uses a greedy method to extract the overall best semantic representations from the packed representation. First, we enumerate the semantic triples (arg1 type, semantic relation, arg2 type) from the packed representation and then score them using a scoring function (explained below). For example, the semantic triples for the packed representation shown in Figure 4.4 are (ignite-3a[Burn] instrument spark-plug-8a[Living-Entity]), (ignite-3a[Burn] instrument spark-plug-8a[Device]), ..., (its-7a[it#prn] has-part spark-plug-8a[Living-Entity]). Then, the highest-scored triple is selected, and the ones incompatible with the selected triple are discarded. For example, if

(ignite-3a[Burn] instrument spark-plug-8a[Living-Entity]) is selected, (ignite-3a[Burn] instrument spark-plug-8a[Device]) is discarded because spark-plug-8a should have only one semantic type. This process is repeated until no triples remain.

The scoring function is learned from a logistic regression model [60] with the training examples, which are pairs of the semantic triples and their class. The class is positive if all three elements in the semantic triple are correct; otherwise, the class is negative. The number of semantic triples used in training is 21,000. The features used are:

From Ally

1. Type score for the head node by SenseRelate
2. Type ranking for the head node by SenseRelate
3. Type score for the tail node by SenseRelate
4. Type ranking for the tail node by SenseRelate
5. Semantic relation score by Ally

From Prismatic

1. Parse score by Prismatic
2. Type score for the head node by Prismatic
3. Type ranking for the head node by Prismatic
4. Type score for the tail node by Prismatic
5. Type ranking for the tail node by Prismatic
6. Semantic relation score by Prismatic

4.3.3 Experiment

To evaluate the benefit of using Prismatic in the disambiguation of the packed representation, we compared three systems. The baseline system imple-

ments the traditional pipeline approach, which selects the highest-scored interpretations at each step. The two other systems are based on the packed representation to delay ambiguity resolution and then use the method explained in Section 4.3.2.4 to select the overall best semantic representations. The systems differ only in the features of the scoring function. The second system(Ally) uses only the features from Ally, while the third system (Ally+Prism) uses all features.

We collected eight news articles about NFL games which consist of 115 sentences in total, and then manually formulated their gold standard semantic representations (GS rep). We used five documents (71 sentences) for training and three documents (26 sentences) for testing.

We used the following metrics to measure the quality of the semantic representations produced by the three systems:

$$
\begin{aligned}
TypePrecision &= \frac{number\ of\ the\ correct\ types}{number\ of\ words\ considered\ by\ the\ system} \\
TypeRecall &= \frac{number\ of\ words\ in\ GS\ rep\ whose\ type\ is\ correctly\ predicted}{number\ of\ words\ in\ GS\ rep} \\
RelPrecision &= \frac{number\ of\ the\ correct\ semantic\ relations}{number\ of\ relations\ considered\ by\ the\ system} \\
RelRecall &= \frac{number\ of\ semantic\ relations\ in\ GS\ rep\ correctly\ predicted}{number\ of\ semantic\ relations\ in\ GS\ rep}
\end{aligned}
$$

4.3.4 Discussion

Table 4.4 shows the result; first, Ally outperforms the baseline overall (except for the recall in semantic relation assignment), indicating that jointly

resolving the ambiguities could be beneficial even without an external knowledge resource. Second, Prismatic's contribution (Ally+Prism) is partially beneficial, improving only the performance of semantic relation assignment but degrading the performance of type assignment.

We hypothesize that a primary reason for Prismatic's minimal contribution may stem from Prismatic's lack of semantic information. For example, we introduced the weight function in the candidate type scoring formula (Section 4.3.2.2) to rule out the Prismatic Frames that contain the input words used as a different sense. This weight function might over-penalize a highly polysemous words (a word with many word senses) by taking the number of the senses in the denominator. If Prismatic frames contain the word sense information, the correct Prismatic frames might be more accurately retrieved without relying on the weight function.

To address this problem, we are developing an algorithm to include more semantic information in Prismatic. Figure 4.21 shows an example of the Prismatic Frame about *rush (in the football game)* produced by our algorithm. This frame contains semantic information such as *rush is a kind of* MOVE and *the objective of rush is to score a touchdown*. This semantic information would be more suitable for resolving semantic ambiguities.

We explain our algorithm step-by-step, with an example of building a frame about the lexeme, *"rush"*.

Step 1. Gather sentences containing the target lexeme.

	Type Precision			Type Recall		
	Correct#	Total#	Ratio	Correct#	Total#	Ratio
Base	227	272	83.45%	227	275	**82.54%**
Ally	227	270	**84.07%**	227	275	**82.54%**
Ally+Prism	219	268	81.71%	219	275	79.63

	Relation Precision			Relation Recall		
	Correct#	Total#	Ratio	Correct#	Total#	Ratio
Base	71	135	52.59%	71	148	47.97%
Ally	69	125	54.40%	68	148	45.94%
Ally+Prism	74	124	**59.67%**	74	148	**50.00**

Table 4.4: Precision and recall for type assignment and semantic relation assignment. The baseline system is based on the traditional pipeline approach. Ally delays resolving ambiguities to jointly resolve them at the last step. Ally+Prism uses Prismatic for disambiguating the packed representation.

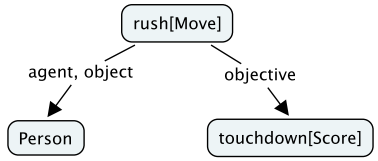


Figure 4.21: Semantic frame about *"rush (in the football game)"*

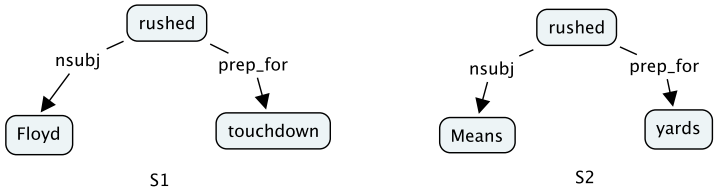
The following shows the example sentences, containing the lexeme *"rush"*, drawn from the news articles about NFL games.

S1: William Floyd rushed for three touchdowns, moving the 49ers one victory from the Super Bowl.

S2: San Diego's Natrone Means rushed 24 times for 139 yards, including a 24-yard touchdown run in the third quarter.

Step 2. Generate Prismatic Frames by applying frame cuts.

This step produces Prismatic Frames by applying frame cuts to the parse trees of the sentences acquired from Step 1. For S1 and S2, this step produces the following:

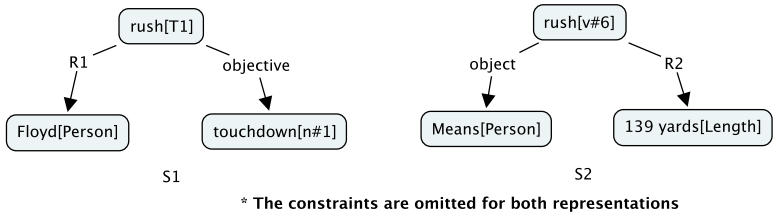


Step 3. Produce the packed representation for the Prismatic Frames.

This step converts the Prismatic Frames into the packed representation using Ally, as shown below.

Step 4. Combine the packed representations to extract the overall best representation.

All these packed representations represent parts of the same semantic knowledge about the target lexeme. Therefore, if the same semantic repre-



sentations appear redundantly across multiple packed representations, we can hypothesize that these representations are likely to be more correct than non-redundant ones. This observation is similar to the one used in Algorithm 2 (Section 4.1.2), which combines redundant sentences.

Similar to Algorithm 2, Step 4 identifies the redundant interpretations across multiple packed representations and then combines them to increase their confidence scores. For example, if we assume that T1 in S1 has a candidate type, MOVE, and that R1 has a candidate semantic relation, *object*, the confidence scores of MOVE and *object* in (rush[Move] object Person) would be increased because the triple appears in both representations. After all packed representations are combined into a single representation, the overall best representation is extracted as in Algorithm 2. Figure 4.21 shows the result of this step.

4.3.5 Summary

Our evaluation shows that our initial approach to using Prismatic is partially beneficial for disambiguating the packed representation. To further improve Prismatic, we presented a future project that aims to add more semantic information into Prismatic.

4.4 Related Work

Several representation schemes have been proposed to compactly represent multiple candidate semantic representations. We discuss two types of compact representations in Sections 4.4.1 and 4.4.2. In Section 4.4.3, we present the work related to disambiguating the packed representation. Finally, in Section 4.4.4, we present the system architectures that attempt to overcome the shortcoming of the traditional pipeline approach.

4.4.1 Packed Representation

The idea of succinctly representing multiple interpretations has been explored by several researchers. For example, the Parc packed representation [98] [35] uses logical formulas to denote different choices of alternative interpretations and treats the disambiguation task as the propositional satisfiability problem. Core Language Engine [4] introduces two types of packing mechanisms. First, they propose a representation, called a quasi logical form, that allows the underspecification of several types of information, such as anaphora references, ellipsis, or semantic relations [5]. They also propose a packed quasi logical form [4] to compactly represent the derivations of alternative quasi logical forms.

Our packed representation contrasts with their representations in several ways. First, our representation is based on the graphical representation. Even though the expressive power of the graphical representation is strictly weaker than full first-order logic [98] (e.g., it is unable to represent negation),

the graphical representation allows the system to employ more powerful reasoning schemes to disambiguate the packed representation (e.g., graph matching in Section 4.1).

Second, our representation explicitly distinguishes different types of (ambiguity) constraints, which allows the system to treat each constraint in a customized way. Finally, our representation includes confidence scores to represent the degree of confidence. It is non-trivial to include numerical scores in the logical framework.

For some NLP tasks, such as parsing and machine translation, several methods have been proposed to compactly maintain multiple candidates. Parse Forest [140] compactly represents multiple candidate phrase-structured parse trees. It is simply a representation of the chart used in the CFG parsing algorithm. Hypergraph [78] generalizes several packed representation (e.g., packed forest structure). Such a structure has been widely used in various applications, such as forest scoring for parsing [70] and the forest-based Statistical Machine Translation model [102].

4.4.2 Underspecified Representation

The previously mentioned representations and our packed representation have one feature in common: they represent a set of complete alternative interpretations of a text. Another class of compact representations, called *underspecification*, has been extensively studied as a formal representation of ambiguous sentences. These representations include hole seman-

tics [18], underspecified discourse representation semantics [123], minimal recursion semantics [34], and dominance constraints [41]. Rather than packing fully-represented candidate interpretations, these representations specify fragments of interpretations, which are unambiguously interpreted, along with constraints on the way the fragments are combined. Different combinations, therefore, correspond to different interpretations. The underspecified representations have generally been focusing on specific types of ambiguities, such as scope ambiguity [18] [41] [34] or discourse relations [123] [122].

4.4.3 Disambiguating Packed Representations

Despite the extensive study of compact representations, the methods to disambiguate them has been far less explored. Riezler et al. [125] and Geman and Johnson [56] use a packed representation to train their parsers on a training corpus and uses the learned statistics to disambiguate the packed representations. Yeh et al. [150] use a hand-built knowledge base to resolve word sense and semantic role ambiguities to disambiguate from a list of candidate interpretations. This dissertation presents other sources of information – redundancy across multiple texts, a semantically annotated corpus, and an automatically constructed knowledge resource.

4.4.4 System Architectures

Several architectures have been proposed to improve the pipeline architecture. Sutton and McCallum [137] and Wellner et al. [145] maintain a

beam of the n best interpretations in the pipeline architecture. However, their pipeline is much shorter than ours, consisting of only two components. Finkel et al. [53] use sampling over the distribution of alternative interpretations at each stage of the pipeline and then passes the sampled data on to the next component. As with our approach, the Parc packed representation [35] and CLE [4] use the packed representation in the pipeline, though both, at some stages, unpack them and re-pack the processed result.

Joint learning and inference have been studied to overcome the limitations of the traditional pipeline approach. This approach generally unifies two closely related tasks (e.g., WSD/SRL [101], parsing/SRL [121], parsing/NER [52]) to learn the functional dependency between the candidate interpretations in the two tasks.

4.5 Summary

Knowledge integration can improve text interpretation by exploiting knowledge from a variety of information. One challenge in this approach is delaying ambiguity resolution during NLP tasks so that knowledge integration can perform it. To address this challenge, we developed the *packed representation* to allow the system to efficiently maintain a myriad of candidate interpretations. Then, knowledge integration commits to an interpretation at the end of the pipeline by aggregating various sources of information. We particularly explored three sources of evidence – redundancy across multiple texts, OntoNotes (a semantically annotated corpus), and Prismatic (an au-

tomatically built knowledge resource). For redundancy and OntoNotes, our evaluation shows that knowledge integration can significantly improve the accuracy of text interpretation by using the packed representation.

Chapter 5

Future Work

In this chapter, we discuss several directions of our future work on knowledge integration. Sections 5.1 through 5.3 discuss the extensions to our work presented in Chapters 3 and 4. Specifically, Section 5.1 discusses the future work about flexible semantic matching; section 5.2 discusses the work about using background knowledge bases for text understanding; section 5.3 discusses more sources of evidence for disambiguating the packed representation; and section 5.4 discusses other future work. Finally, Section 5.5 presents several applications of knowledge integration and discusses the application-oriented evaluation for machine reading.

5.1 Aligning Semantic Representations

In Section 2.3.3.1, we presented several cases in which the semantic representations could differ even though they express the same content. To resolve these mismatches, these cases require sophisticated methods, such as our flexible matching (Chapter 3). We review each of these cases and present how they could be addressed.

5.1.1 Resolving Granularity Mismatch

In Section 3.2.1, we presented the domain-independent patterns for resolving granularity mismatches. One common case, which is not addressed in this dissertation, is lexical-level granularity mismatch. To illustrate, consider the sentence: *"the blood is oxygenated"*. The lexical item, *oxygenation*, represents two complex series of events: the movement of the oxygen from the air to an object and the chemical combination of the oxygen and the object. When these events are described, they should be properly aligned with *oxygenation*.

To resolve lexical-level granularity differences, the system has to know the meaning of the lexical items. For this requirement, several methods could be useful. For example, semantic decomposition, reviewed in Section 3.4.2, attempts to represent the meaning of the lexical items in a canonical way. Wilks et al. [146] also discuss producing the formal representations of the dictionary definitions.

The granularity mismatch may also occur at the representational level. For example, consider representing the following simple sentence: *"The body has arms"*. The Component Library can represent this sentence simply by using its primitive relation, *has-part* – i.e., (body has-part arm). The NLP software, however, may produce a more fine-grained representation by mapping *has* to a concept – possibly, a concept representing the state of *having*, thereby producing (Have base body) (Have object arm) [1]. Because these cases

[1]Even though this is a semantic interpretation problem, knowledge integration should be able to address these types of knowledge representations.

occur often, manually identifying them is challenging. Automated mining techniques [149] could be useful for this task.

5.1.2 Viewpoint Difference

The same content could be expressed differently because of differences in viewpoint, for example, using a simile and metaphor. Consider the following texts:

(1) *Blood circulation is like electrical circulation. Just as electrons carry electrical charge a battery to electronic devices, the blood carries oxygen from the heart to the organs (simile).*

(2) *The white cell fights against viruses (metaphor).*

Building a formal model from (1) requires more than merely combining the individual sentences. The analogical relationship should be identified between blood circulation and electrical circulation; then, based on the analogical relationship, the knowledge about the electrical circulation should be used to elaborate the blood circulation. (2) also requires similar processing even though the base domain is unspecified. The system should identify the analogical relationship between the activity of the white cell and *the war*, and the knowledge about the war should be properly used to interpret (2). Several approaches have been proposed to interpret the metaphor [49] [107] and simile [8]. Agerri [1] uses the metaphorical interpretation in textual entailment.

A temporal relationship is often described differently because of the differences in viewpoint. Consider the following sentences:

(1) *The heart pumps blood to the lung, and then the blood picks up the oxygen.*

(2) *The heart pumps blood to the lung for the blood to pick up the oxygen.*

(3) *The heart pumps blood to the lung to enable the blood to pick up the oxygen.*

All three sentences represent the same phenomenon, but they differently describe the relationship between the pumping of blood and the acquisition of oxygen. (1) describes it as the temporal relation. (2) describes it as a teleological relation – i.e., the purpose of the pumping is to achieve the acquisition of the oxygen. (3) describes it as *enablement* – i.e., the pumping makes it possible to acquire blood. Because of these differences, different semantic relations from the Component Library are used to represent the temporal relationship – *next-event* for (1), *causes* for (2), and *enables* for (3). It is important for knowledge integration to identify these differences and resolve them properly.

5.1.3 Distinguishing Different Contexts

Context-dependent assertions should be distinguished rather than blindly integrated. For example, different political parties may give opposing opinions, or consumers may give conflicting reviews about products. These differences should be properly distinguished to make the resulting knowledge base consistent.

The CYC knowledge base addresses this problem by splitting the knowl-

edge base into multiple partitions called microtheories. Each microtheory contains only a consistent and coherent set of assertions, and the conflicting information is stored in different microtheories. For example, the following conflicting assertions – *"The earth is flat"* and *"The earth is round"* – would be stored in different microtheories. The former would be stored in the microtheory for reasoning about everyday life, whereas the latter would be stored in the microtheory about astronomy.

The partitioning approach, however, fails to capture the differences between the conflicting assertions. It would be more useful to represent how the assertions differ by contrasting the opposing aspects. For example, consider combining the following sentences: *"The heart has four chambers (in humans)"* and *"The heart has three chambers (in amphibians)"*. Rather than storing them separately, it would be more effective to represent that the hearts of humans and amphibians have a different number of chambers (along with the actual numbers).

Several previous projects are related to handling context-dependent assertions. Sentiment analysis is concerned with classifying the texts into sentiment categories. Taylor et al. [138] present a method for selecting an appropriate microtheory to store new information in the CYC knowledge base. However, most of these projects only focus on classifying texts into different contexts rather than on identifying the contrasting features among the opposing assertions.

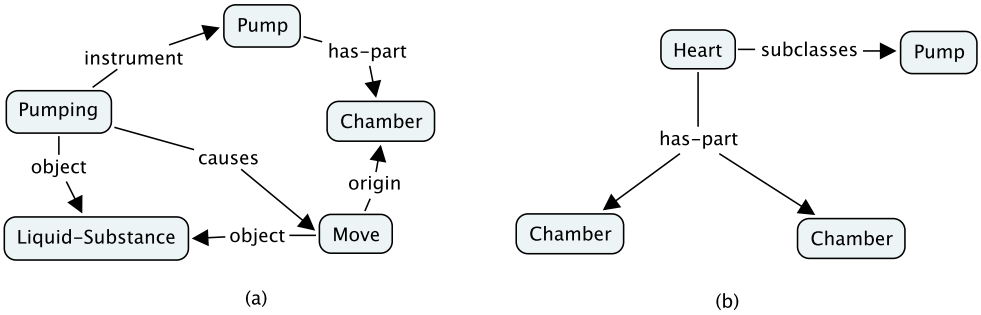


Figure 5.1: (a) The one chamber pumping model; (b) The semantic representation that describes the heart with the two chambers

5.2 Inferring Unspecified Information

Text understanding requires a vast amount of background knowledge to reveal implicit information in texts. To address this problem, Kleo uses the contents learned from previous reading and the axioms in the Component Library as its background knowledge. Then, it performs spreading activation to retrieve the knowledge relevant to the texts from the knowledge base (see the *Elaborate* algorithm in Section 2.3.3.1). In this section, we discuss several future projects in using the background knowledge base for text understanding.

5.2.1 Adapting Background Knowledge Representation

One type of knowledge that *Elaborate* retrieves is the formal model of general concepts. For example, (a) in Figure 5.1 shows the PUMPING model used by Kleo to understand texts in the heart domain.

The model, however, often fails to fit with the texts, requiring that it be modified to improve matching. For example, consider aligning the pump-

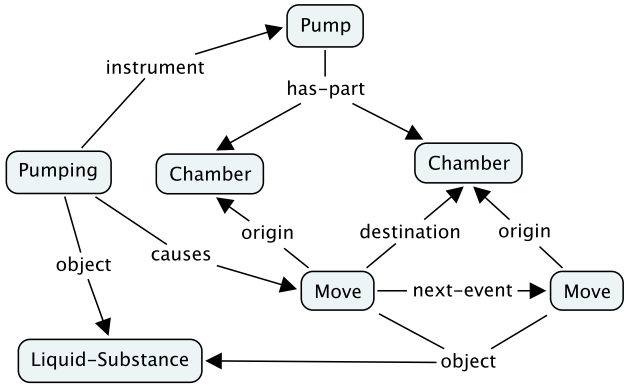


Figure 5.2: The two chamber pumping model adapted from the one chamber pumping model

ing model with the semantic representation ((b) in Figure 5.1) of the sentence *"The heart in the fish has two chambers"*. If the original model is naïvely aligned (i.e., graph-matched) with the semantic representation, only one chamber would be elaborated while the other one is left unchanged. To elaborate the sentence correctly, the original model should be modified to the two-chamber pumping model (shown in Figure 5.2), and the new model should be aligned with the sentence. This task is important, particularly for the models to have broad coverage.

Case-based reasoning addresses a similar problem, which should adapt the previous problems and their solutions to the new task. For example, Smyth and Keane [131] use the manually formulated adaptation knowledge to modify the representations of the previous tasks and solutions.

5.2.2 Reasoning for Inference

Kleo uses simple reasoning, spreading activation, to retrieve the knowledge relevant to the text from the knowledge base. Incorporating more reasoning methods would allow the system to make better inferences. For example, the sentence, *"The blood travels from the heart to the lung and then picks up the oxygen"*, omits the location of picking up the oxygen, but this could be inferred if the system simulates the events described in the text.

Several approaches have been proposed to perform various types of reasoning. Fuzzy logic [153] allows a system to reason with vague concepts, which are often described in natural texts. Narayanan [107] simulates the actions described by metaphors. Tacitus [65] uses abductive reasoning to make an overall coherent interpretation of the text. Several logical frameworks, called Natural Logic [90] [128], model certain types of natural language expressions and allow the system to reason with the expressions.

5.3 Other Sources of Evidence for Text Interpretation

In Chapter 4, we presented the packed representation to delay ambiguity resolution and explored three sources of evidence for disambiguating the packed representation. In this section, we introduce other sources of evidence.

5.3.1 Coherence of Paragraphs

A paragraph generally organizes sentences so that an idea is coherently delivered [154]; therefore, measuring the extent to which the candidate

interpretations of the consecutive sentences are coherent could help identify the correct interpretations. In general, background knowledge is needed to measure the level of coherence among a set of interpretations. Zadrozny and Jensen [154], for example, propose using dictionary definitions as background knowledge to measure the coherence of the text interpretation. A practical computational model of coherence, however, has not yet emerged.

5.3.2 Reading Preliminary Texts

In Section 4.1, we showed that joint interpretation of multiple texts could improve the accuracy of interpretation by exploiting redundancy across multiple texts. This positive result indicates that if the preliminary texts (e.g., dictionary or textbooks for children) could be reliably interpreted, they could be useful for reading more complicated texts (e.g., college-level textbooks).

5.3.3 Other External Knowledge Resources

In Section 4.1, we evaluated two external knowledge resources, OntoNotes and Prismatic. Other knowledge resources might also be useful for disambiguating the packed representations, such as linguistic knowledge resources [51] [129], common-sense knowledge bases [130] [85], and other automatically constructed knowledge resources [32] [88].

5.4 Other Future Projects

In this section, we present future projects on the following tasks: managing the belief of the extracted information, selecting the informative texts, and dealing with other types of texts.

5.4.1 Belief Management

Sophisticated belief management is important in machine reading systems to manage the uncertainty of the learned knowledge because of the errors of NLP. For example, the system should be able to retract the learned knowledge if the knowledge is found to be wrong in subsequent reading.

We proposed one belief management method based on the packed representation in which the system maintains multiple candidates and narrows down these choices as more evidence is gathered. The disadvantage to this architecture is that once the system commits to an interpretation, it is never retracted from the knowledge base. The ability to retract an interpretation is crucial because the committed interpretations could be wrong.

Nell [24] incorporates a belief management facility. In Nell, the extracted knowledge is stored along with its confidence score in a temporary pool and then is promoted to the final knowledge base when the score has sufficiently increased. As in our architecture, however, the final knowledge base monotonically increases, never allowing the knowledge to be withdrawn.

5.4.2 Content Selection

It is important to identify uninformative sentences to filter out their semantic representations. For example, consider the following text used in our experiment, in which only a few sentences deliver useful knowledge about blood circulation:

> *"What is the heart? Why is it so important? What does it do in the body? You probably think you know what the heart looks like. But you are probably wrong. The heart does not look very much like the shapes people draw on Valentine's Day. And it certainly isn't flat like a paper valentine. When you pledge allegiance to the flag, you place your hand over the left side of your chest. Do you know why? That is supposed to be where the heart is."*

A related and interesting work is targeted reading, in which the system actively finds informative texts to fill the gap of missing knowledge, rather than passively receiving texts. Davis and Buchanan [36] perform a similar task, which identifies the missing contents in the knowledge base using meta-knowledge. Our packed representations could also be useful for this task, informing the system of what kind of knowledge is needed for ambiguity resolution.

5.4.3 Other Types of Texts

In this dissertation, we focused only on the technical texts describing the concepts and their relations. One important direction of future work is

to explore other types and genres of texts. For example, micro-texts in social networks often express subjective opinions or sentiments; a primary knowledge integration task for these texts would be to contrast the opposing views. Children's stories also present unique challenges as they oftentimes describe imaginary worlds. As such, the assertions in the background knowledge base may need to be changed to deal with unreal things (e.g., talking animals).

5.5 Applications and Wide-Scale Experiment

Finally, we present several applications of knowledge integration (Sections 5.5.1 $\sim$ 5.5.3) and a plan for a wide-scale experiment (Section 5.5.4).

5.5.1 Competitive Intelligence

One significant task of a competitive intelligence system (e.g., [152]) is to gather and analyze information to support decision-making in business, using information sources such as news articles and company reports. For example, news articles may report the recent activities of rival companies, and annual company reports may specify crucial facts about the company's recent performance. Given these texts, the system should be able to extract useful pieces of information and coherently combine the extracted information (e.g., displaying the event snippets temporally).

5.5.2 Social Analysis

Social media (e.g., Facebook, Twitter) has become a criticial piece for business companies to better understand product launches, consumer profiles, and market campaign effectiveness. Knowledge integration and the NLP technologies could be applied to combine information about a single or group of consumer(s) from multiple social media sites. This would, for example, enable the system to automatically build a complete understanding of a consumer's interests, intent, values, and so on.

5.5.3 Information Management

The information management system maintains various types of information, such as documents, images, presentation files, and databases. Knowledge integration could be useful in their management by combining semantically related data from these disparate sources, which, in turn, would allow the system to produce richer and more coherent information, enabling higher quality analysis and reporting.

5.5.4 Application-Oriented Evaluation

In this dissertation, we intrinsically evaluated the output knowledge base by measuring its cohesiveness (Chapter 3) and correctness (Chapter 4). Equally important is the extrinsic evaluation, which measures the impact of the knowledge base built by our approach on applications such as question-answering.

Chapter 6

Conclusion

This dissertation studies the problem of knowledge integration, a task of combining knowledge snippets into a single coherent knowledge base. In this final chapter, we revisit the goals of our research and summarize our contribution (Section 6.1). Then, we present the lessons we learned from this research (Section 6.2) and conclude this dissertation (Section 6.3).

6.1 Goals Revisited and Summary of Our Contribution

In this section, we revisit the two goals of this dissertation and summarize our contribution for each goal.

6.1.1 Coherently Combining Knowledge Snippets

The first goal of this dissertation was to show that knowledge integration improves the reasoning power of the output knowledge base. To illustrate, we wanted to build a machine reading system equipped with sophisticated knowledge integration facilities and to evaluate the system by measuring the quality of its output knowledge base.

For this goal, we built an end-to-end reading system, Kleo. Kleo per-

forms two types of knowledge integration: sentence-to-sentence knowledge integration (SKI) and text-to-text knowledge integration (TKI). SKI combines the sentences within the same text to build the formal representation of the text. TKI combines the outputs of SKI across multiple texts.

SKI consists of two steps: *Stitch* and *Elaborate*. *Stitch* combines the semantic representations of the sentences using our matcher, and *Elaborate* augments the stitched result with the knowledge base (the contents acquired from previous reading and the axioms in Component Library). *Elaborate* performs spreading activation through the knowledge base to retrieve background knowledge relevant to the text and then aligns the retrieved knowledge with the text.

TKI also performs two steps: *Partitioning* and *KB Update*. *Partitioning* splits the output of SKI into multiple coherent units called K-units. The purpose of *Partitioning* is to improve the scalability of the knowledge base update by reducing the size of knowledge representations. *KB Update* updates the knowledge base with new K-units.

The core component of SKI and TKI is our flexible matcher, which resolves the common types of granularity mismatches among the knowledge snippets. To develop this matcher, we identified four common types of granularity mismatches – filtering, generalization, abstraction, and co-reference across granularity difference – and then, based on these types, developed general patterns for resolving the granularity mismatches.

To evaluate our knowledge integration methods and because cohesive knowledge bases are computationally useful in general, we measured the density of the knowledge base built by Kleo. Our evaluation shows that Kleo's knowledge integration, especially, the flexible graph matcher and *Elaborate*, is effective in increasing density without degrading the correctness of the knowledge base.

6.1.2 Applying Knowledge Integration to Text Interpretation

The second goal of this dissertation was to show that knowledge integration improves text interpretation by exploiting a variety of information, such as other texts and external knowledge bases. We particularly focused on two tasks – delaying ambiguity resolution during the NLP tasks in the pipelined system to avoid aggressive pruning and exploring the sources of evidence for ambiguity resolution. Then, we evaluated our approach by measuring the quality of the semantic representations produced by our approach.

To delay ambiguities in the pipelined system, we developed the packed representation to efficiently manage a myriad of candidate interpretations produced by the NLP tasks. The packed representation also explicitly represents the relationship among the candidates to effectively prune away implausible candidates. The ambiguities in the packed representation are then resolved by knowledge integration, which considers various sources of evidence.

We explored three sources of evidence: redundancy across multiple texts, OntoNotes (a semantically annotated corpus), and Prismatic (a knowl-

edge base automatically constructed from texts). Our evaluation shows that, for redundancy and OntoNotes, our approach significantly improves the quality of the semantic representations using the packed representation.

6.2 Lessons Learned

First, without sophisticated knowledge integration, the knowledge snippets produced by machine reading systems would be often incorrectly aligned, producing a fragmented or incorrect knowledge base. The difficulty of NLP compounds this problem because it produces fragmented or incorrect semantic representations that may misguide the subsequent knowledge integration component.

Second, one major difficulty of knowledge integration is the variety of textual forms that express the same content. Because of these representational differences, simple alignment (e.g., aligning the maximal common subgraph) often fails to combine the knowledge snippets. More sophisticated methods should be developed to resolve the mismatches.

Third, the variety of textual forms, however, could be advantageous to text interpretation because it allows the system to combine the results of processing the different forms, which may jointly yield better semantic representations than processing them independently. Our algorithm for using redundancy shows the promise of this approach.

Fourth, supplying background knowledge is still a significant problem

in text understanding. We attempted to address this problem by using other texts and external knowledge resources, OntoNotes and Prismatic. Recently, various types of knowledge resources have been developed, such as semantically annotated corpora, Semantic Web, and linguistic knowledge resources. It would be beneficial future research to develop sophisticated methods for using those resources in text understanding and to identify their strengths and weaknesses.

Fifth, the NLP component tasks have received much more attention than end-to-end reading systems. As this dissertation shows, however, the pipeline architecture, which has been widely used as the architecture of machine reading systems, is unsuitable, and a better architecture should be developed to make a system robust to NLP errors and to reliably commit to an interpretation. The key problem in this research is to make the individual components communicate with one another to resolve ambiguities more reliably. In this dissertation, we proposed one approach based on the packed representation.

Sixth, our prototype system, Ally, improved interpretation accuracy by using redundancy and OntoNotes but still prunes away many correct interpretations at the final decision point. This shows that accuracy can be further improved by using more evidence such as other texts and external knowledge resources.

Finally, a system-level evaluation methodology should be devised to evaluate machine reading systems. As the NLP research has focused on indi-

vidual component tasks, most NLP evaluation has been measuring the performance gain at the component level. System-level evaluations are also crucial to determine whether the performance gain in the components can translate to improving the overall end-to-end performance.

6.3 Closing Words

Machine Reading is a promising solution to the Knowledge Acquisition Bottleneck, but still requires addressing several challenges. This dissertation studied one of the challenges, knowledge integration – a task of combining knowledge snippets into a coherent whole. We were concerned with two hypotheses: 1) knowledge integration improves the reasoning power of the knowledge base and 2) knowledge integration improves text interpretation. To evaluate these hypotheses, we built two proof-of-concept reading systems, Kleo and Ally. Kleo, equipped with sophisticated knowledge integration facilities, improves the cohesiveness of the output knowledge base – a cohesive knowledge base is computationally useful in general. Ally, based on knowledge integration, improves the accuracy of text interpretation.

The conclusion is that knowledge integration is computationally tractable to produce coherent knowledge bases and merits further investigation. We also conclude that knowledge integration relieves the burden of full NLU, an unsolved problem in AI, by drawing and effectively combining information from multiple texts.

Appendices

Appendix A

Ten paraphrases of the heart text

This appendix presents the text submitted to Amazon Turk (text1) and the nine paraphrases (text2 $\sim$ test10).

> Text1: Hearts pump blood through the body. Blood carries oxygen to organs throughout the body. Blood leaves the heart, then goes to the lungs where it is oxygenated. The oxygen given to the blood by the lungs is then burned by organs throughout the body. Eventually the blood returns to the heart, depleted of oxygen.

> Text2: Blood travels through the heart to the lungs, where it is oxygenated. It then travels to the various organs throughout the body. The organs use the oxygen in the blood, so that by the time the blood returns to the heart, it is depleted of oxygen.

> Text3: Blood is essentially the transportation mechanism for the delivery of oxygen and other essential nutrients in the body. It can be compared to the train and bus systems in cities where these modes of transport circulate continuously amidst various terminals, stops, depots and storage or refuelling barns. The circulation of blood is powered by its being pumped through the heart. The

blood then takes on oxygen in the lungs. This oxygen is delivered to organs throughout the body where it is used as fuel. Blood returns to the heart and the transportation loop begins again.

Text4: The heart pumps blood around the body, carrying oxygen to various organs. Blood first leaves the heart to go to the lungs, where it picks up oxygen before being sent to the rest of the body's organs. Eventually, when the oxygen has been taken up by other organs, the blood comes back to the heart.

Text5: The heart begins to pump blood into the body. The blood first travels to the lungs, where it picks up oxygen. The blood will then be deposited into the organs, which burn the oxygen. The blood will then return to the heart, where it will be lacking oxygen, and start over again.

Text6: Hearts are motors that drive blood through your system. This blood brings oxygen to all your organs. After being pumped out of the heart, the blood is sent to the lungs, where it picks up oxygen. After leaving the lungs, the blood brings the oxygen to organs throughout the body to use as fuel. After delivering its oxygen, the blood makes its way back to the heart.

Text7: The heart's main purpose is to pump oxygenated blood throughout the body. The heart sends the blood to the lungs to be oxygenated and then it is sent+ on to the other vital organs in the body. When those organs have depleted the oxygen the blood is then sent back to the heart where the cycle begins anew.

Text8: The heart is the main organ in the circulatory system. Its job is to move blood through the body. Blood carries oxygen from the lungs to the other organs in the body, where the oxygen is used by the organs. Then the blood returns back to the heart and eventually back to the lungs where it can restore its oxygen.

Text9: Circulation is achieved by the rhythmic beating of the heart. After leaving the heart, the oxygen-depleted blood is oxygenated in the lungs. The freshly oxygenated blood exits the lungs and carries oxygen to the organs of the body. The organs throughout the body then utilize the oxygen given to the blood by the lungs. Oxygen-depleted blood finally returns to the heart, ready to begin the circuit again.

Text10: The heart pumps blood throughout the body and in turn the blood carries oxygen to every organ. When the blood leaves the heart it travels through the lungs to oxygenate them. This oxygen is then used by the body's organs. Finally the blood finds it way back to the heart with no oxygen left in it.

Appendix B

Converting packed representation to Alchemy statements

This appendix describes the translation of our packed representation (Section 4.2.2) into the Alchemy statements.

PARSEXOR(p_1, p_2, ..., p_n)

If ϕ [1] is included in PARSEXOR,

$$p_i \rightarrow !(p_1 \wedge ...p_{i-1} \wedge p_{i+1}... \wedge p_n). \text{ for all } i \in 1, .., n$$

Otherwise,

$$p_i \leftrightarrow !(p_1 \wedge ...p_{i-1} \wedge p_{i+1}... \wedge p_n). \text{ for all } i \in 1, .., n$$

p_i is a binary variable that takes 1 (if the corresponding dependency triples are true) or 0 (otherwise). If ϕ is included in PARSEXOR, p_i cannot be confirmed even though the others are found to be false. If ϕ is not included, the above statement expresses a mutually exclusive relationship among p_is: p_i is true if and only if the other variables are false.

[1] ϕ indicates the empty set. PARSEXOR(p_1, p_2, ..., ϕ) means that if p_i is correct, the others are wrong, but does not mean that if all but p_i are incorrect, p_i is correct. See Section 4.2.3.1.

$$
\begin{aligned}
1\ \mathit{sent} &\rightarrow p_i \text{ if the corresponding dependency triples} \\
&\qquad \text{appear in the top-scored parse} \\
.3\ \mathit{sent} &\rightarrow p_j \text{ otherwise}
\end{aligned}
$$

The above statements express the preference for the top-scored candidate parse *a priori*. 1 and .3 are the weights. If there is no evidence to override this preference, the system will choose the dependency triples from the top-scored parse. *sent* is a predicate defined as evidence.

TYPEXOR($\mathbf{w}_i$, (t_1 s_1), (t_2 s_2) , ... , (t_n s_n))

$$
\begin{aligned}
& ws\_w_i(< variable >!) \\
\log s_i \qquad & sent \rightarrow ws\_w_i(t_i) \text{ for all } i \in 1, .., n
\end{aligned}
$$

$ws\_w_i(t_j)$ represents that t_j is the sense of the word w_i. The first rule (defined in the header) asserts that there should be only one sense for w_i. The second rule specifies the weight ($\log s_i$) for each candidate sense.

RELXOR($\mathbf{w}_i$ $\mathbf{w}_j$ (r_1 s_1), (r_2 s_2) , ... , (r_n s_n))

$$
\begin{aligned}
& rel\_w_i\_w_j(< variable >!) \\
\log s_i \qquad & sent \rightarrow rel\_w_i\_w_j(r_i) \text{ for all } i \in 1, .., n
\end{aligned}
$$

$rel\_w_i\_w_j(r_k)$ represents that r_k is a semantic relation connecting w_i and w_j.

PARSEDEP(p_1, p_2)

$$p_1 \rightarrow p_2.$$

This rule states that if p_2 is false, p_1 should be false.

DRV($\mathbf{R}_1$, (p_1, t_{11}, t_{12}, (r_{11} s_{11}), ..., (r_{1m} s_{1m})), ..., (p_n, t_{n1}, t_{n2}, (r_{n1} s_{n1}), ..., (r_{nm} s_{nm})))

$$\log s_{ij} \qquad p_i \wedge ws\_w_1(t_{i1}) \wedge ws\_w_2(t_{i2}) \rightarrow rel\_w_1\_w_2(r_{ij})$$

$$\text{for all } i \in \{1, .., n\} \setminus \{NIL\}$$

$$p_a \vee ... \vee p_z \leftrightarrow !rel\_w_1\_w_2(NIL).$$

The first rule directly represents the derivation relationship except for the NIL candidate. It represents that if p_i, t_{i1} and t_{i2} are true, r_{ij} is a correct semantic relation with the weight $\log s_{ij}$. In the second rule, $p_a, ..., p_z$ denote a parse fragment containing a dependency triple connecting w_1 and w_2. The rule says that any of $p_a, ..., p_z$ is true if and only if a semantic relation exists between w_1 and w_2.

Bibliography

[1] Rodrigo Agerri. Metaphor in textual entailment. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 3–6, 2008.

[2] Eneko Agirre and Oier Lopez de Lacalle. On robustness and domain adaptation using SVD for word sense disambiguation. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 17–24, 2008.

[3] James Allen. *Natural Language Understanding*. Benjamin/Cummings, 1987.

[4] Hiyan Alshawi, editor. *The Core Language Engine*. MIT Press, Cambridge, MA, 1992.

[5] Hiyan Alshawi and Richard Crouch. Monotonic semantic interpretation. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics (ACL-92)*, pages 32–39, 1992.

[6] J. R. Anderson. *The Architecture of Cognition*. Havard University Press, Cambridge, MA, 1983.

[7] Michele Banko, Michael J. Cafarella, Stephen Soderland, Matt Broadhead, and Oren Etzioni. Open information extraction from the web.

In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)*, pages 2670–2676.

[8] David Barbella and Kenneth Forbus. Analogical dialogue acts: supporting learning by reading analogies. In *Proceedings of the NAACL/HLT Workshop on Formalisms and Methodology for Learning by Reading*, pages 96–104, 2010.

[9] Ken Barker, Bhalchandra Agashe, Shaw-Yi Chaw, James Fan, Noah Friedland, Michael Glass, Jerry Hobbs, Eduard Hovy, David Israel, Doo Soon Kim, Rutu Mulkar-Mehta, Sourabh Patwardhan, Bruce Porter, Dan Tecuci, and Peter Yeh. Learning by reading: A prototype system, performance baseline and lessons learned. In *Proceedings of 21st National Conference on Artificial Intelligence (AAAI-07)*, pages 280–286, 2007.

[10] Ken Barker, Jim Blythe, Gary Borchardt, Vinay K. Chaudhri, Peter E. Clark, Paul Cohen, Julie Fitzgerald, Ken Forbus, Yolanda Gil, Boris Katz, Jihie Kim, Gary King, Sunil Mishra, Clayton Morrison, Ken Murray, Charley Otstott, Bruce Porter, Robert C. Schrag, Toms Uribe, Jeff Usher, and Peter Z. Yeh. A knowledge acquisition tool for course of action analysis. In *Proceedings of 15th Innovation Application of Artificial Intelligence (IAAI-03)*, pages 43–50, 2003.

[11] Ken Barker, Bruce Porter, and Peter Clark. A library of generic concepts for composing knowledge bases. In *Proceedings of 1st International Conference on Knowledge Capture (KCAP-01)*, pages 14–21, 2001.

[12] Ken Barker and Stan Szpakowicz. Interactive semantic analysis of clause-level relationships. In *Proceedings of the 2nd conference of the Pacific Association for Computational Linguistics (PACLING-95)*, pages 22–30, 1995.

[13] Cosmin Adrian Bejan and Sanda Harabagiu. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL-10)*, pages 1412–1422, 2010.

[14] Steven Bethard and James H. Martin. Learning semantic links from a corpus of parallel temporal and causal relations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers (HLT-Short-08)*, pages 177–180, 2008.

[15] Thomas Bittner and Barry Smith. A taxonomy of granular partitions. In *Proceedings of the International Conference on Spatial Information Theory: Foundations of Geographic Information Science (COSIT-01)*, pages 28–43, 2001.

[16] Branimir Boguraev and Rie Kubota Ando. TimeML-compliant text analysis for temporal reasoning. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 997–1003, 2005.

[17] Branimir Boguraev and Rie Kubota Ando. Effective use of timebank for timeml analysis. In *Proceedings of the 2005 International Conference on Annotating, Extracting and Reasoning about Time and Events*, pages 41–58, 2007.

[18] Johan Bos. Computational semantics in discourse: Underspecification, resolution, and inference. *Journal of Logic, Language, and Information*, 13(2):139–157, 2004.

[19] R. R. Bouckaert. Bayesian network classifiers in Weka. Technical Report 14/2004, Computer Science Department, University of Waikato, September 2004.

[20] Susan E. Brennan, Marilyn W. Friedman, and Carl Pollard. A centering approach to pronouns. In *Proceedings of the 25th Annual Meeting of the Association for Computational Linguistics (ACL-87)*, pages 155–162, 1987.

[21] Joan Bresnan. *Lexical-Functional Syntax*. Blackwell, Oxford, 2001.

[22] Razvan Bunescu. Associative anaphora resolution: A web-based approach. In *Proceedings of the EACL2003 Workshop on the Computational Treatment of Anaphora*, pages 47–52, 2003.

[23] Claire Cardie and Kiri Wagstaff. Noun phrase coreference as clustering. In *Proceedings of the Joint Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 82–89, 1999.

[24] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka Jr., and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of 24th National Conference on Artificial Intelligence (AAAI-10)*, 2010.

[25] Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. Building a Discourse-Tagged Corpus in the Framework of Rhetorical Structure Theory. In Jan van Kuppevelt and Ronnie Smith, editors, *Current Directions in Discourse and Dialogue*, pages 85–112. Kluwer Academic Publishers, 2003.

[26] Daniel Cer, Marie-Catherine de Marneffe, Daniel Jurafsky, and Christopher D. Manning. Parsing to Stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of the 7th Language Resources and Evaluation Conference (LREC-10)*, 2010.

[27] Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-09)*, pages 602–610, 2008.

[28] Nathanael Chambers, Shan Wang, and Daniel Jurafsky. Classifying temporal relations between events. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, 2007.

[29] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings*

of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL-00), pages 132–139, 2000.

[30] Wanxiang Che, Ting Liu, and Yongqiang Li. Improving semantic role labeling with word sense. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-10)*, pages 246–249, 2010.

[31] Timothy Chklovski. Learner: a system for acquiring commonsense knowledge by analogy. In *Proceedings of the 2nd International Conference on Knowledge Capture (KCAP-03)*, pages 4–12.

[32] Timothy Chklovski and Patrick Pantel. Verbocean: Mining the web for fine-grained semantic verb relations. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-04)*, pages 33–40, 2004.

[33] Peter Clark, Phil Harrison, Tom Jenkins, John Thompson, and Rick Wojcik. Acquiring and using world knowledge using a restricted subset of english. In *Proceedings of the 18th Florida Artificial Intelligence Research Society Conference (FLAIRS-05)*, pages 506–511.

[34] Ann Copestake, Dan Flickinger, Carl Pollard, and Ivan Sag. Minimal recursion semantics: an introduction. *Research on Language and Computation*, 3:281–332, 2005.

[35] Dick Crouch. Packed rewriting for mapping semantics to KR. In *Proceedings of 6th International Workshop on Computational Semantics*, 2005.

[36] Randall Davis and Bruce C. Buchanan. Meta-level knowledge. In B. G. Buchanan and E. H. Shortliffe, editors, *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Addison–Wesley Publishing Company, Reading, Massachusetts, 1984.

[37] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence Journal*, (28):127–162, 1986.

[38] AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Y. Halevy. Ontology matching: A machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 385–404. Springer, 2004.

[39] Doug Downey, Oren Etzioni, and Stephen Soderland. A probabilistic model of redundancy in information extraction. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)*, 2005.

[40] David duVerle and Helmut Prendinger. A novel discourse parser based on support vector machine classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL/AFNLP-09)*, pages 665–673, 2009.

[41] Markus Egg, Alexander Koller, and Joachim Niehren. The constraint language for lambda structures. *Journal of Logic, Language, and Information*, 10:457–485, 2001.

[42] Katrin Erk. A simple, similarity-based model for selectional preferences. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, 2007.

[43] Oren Etzioni, Michael Cafarella, Doug Downey, Ana-Maria Popescu, Tal Shaked, Stephen Soderland, Daniel S. Weld, and Alexander Yates. Unsupervised named-entity extraction from the web: An experimental study. *Artificial Intelligence*, 165(1):91–134, 2005.

[44] Jerome Euzenat, Antoine Isaac, Christian Meilicke, Pavel Shvaiko, Vojtch Svtek, Willem Robert Van Hage, and Mikalai Yatskevich. Results of the ontology alignment evaluation initiative 2006. In *Proceedings of the 1st International Workshop on Ontology Matching (OM-2006)*, 2006.

[45] Jerome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer, Heidelberg, 2007.

[46] Brian Falkenhainer, Kenneth D. Forbus, and Dedre Gentner. The structure-mapping engine: algorithm and examples. *Artificial Intelligence Journal*, 41:1–63, November 1989.

[47] James Fan, Ken Barker, and Bruce Porter. Indirect anaphora resolution as semantic path search. In *Proceedings of 3rd International Conference*

on Knowledge Capture (KCAP-05), pages 153–160, Banff, Canada, 2005.

[48] James Fan, David Ferrucci, David Gondek, and Aditya Kalyanpur. Prismatic: inducing knowledge from a large scale lexicalized relation resource. In *Proceedings of the NAACL/HLT Workshop on Formalisms and Methodology for Learning by Reading*, 2010.

[49] Dan Fass. *Processing Metonymy and Metaphor*. Ablex Publishing, Greenwich, Connecticut, 1997.

[50] Gilles Fauconnier and Mark Turner. *The Way We Think: Conceptual Blending and the Mind's Hidden Complexities*. Basic Books, 2003.

[51] Charles J. Fillmore and Collin Baker. *A Frame Approach to Semantic Analysis*. Oxford Univ. Press, 2010.

[52] Jenny Rose Finkel and Christopher D. Manning. Joint parsing and named entity recognition. In *Proceedings of Human Language Technology: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL-09)*, pages 326–334, 2009.

[53] Jenny Rose Finkel, Christopher D. Manning, and Andrew Y. Ng. Solving the problem of cascading errors: approximate Bayesian inference for linguistic annotation pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP-06)*, pages 618–626, 2006.

[54] Noah S. Friedland, Paul G. Allen, Gavin Matthews, Michael Witbrock, David Baxter, Jon Curtis, Blake Shepard, Pierluigi Miraglia, Jürgen Angele, Steffen Staab, Eddie Moench, Henrik Oppermann, Dirk Wenke, David Israel, Vinay Chaudhri, Bruce Porter, Ken Barker, James Fan, Shaw Yi Chaw, Peter Yeh, Dan Tecuci, and Peter Clark. Project Halo: Towards a digital Aristotle. *AI Magazine*, 25(4):29–48, 2004.

[55] Gerald Gazdar, Ewan Klein, Geoffrey, K. Pullum, and Ivan A. Sag. *Generalized Phrase Structure Grammar*. Harvard University Press, Chicago, IL, 1985.

[56] Stuart Geman and Mark Johnson. Dynamic programming for parsing and estimation of stochastic unification-based grammars. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 279–286, 2002.

[57] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

[58] B. J. Grosz, A. K. Joshi, and S. Weinstein. Centering: a framework for modelling the local coherence of discourse. *Computational Linguistics*, 21(2):203–226, 1995.

[59] Aria Haghighi and Dan Klein. Unsupervised coreference resolution in a nonparametric Bayesian model. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, 2007.

[60] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

[61] B. Harrington and S. Clark. Asknet: Automated semantic knowledge network. In *Proceedings of 21st National Conference on Artificial Intelligence (AAAI-07)*, pages 1862–1863, 2007.

[62] Hugo Hernault, Danushka Bollegala, and Mitsuru Ishizuka. Towards semi-supervised classification of discourse relations using feature correlations. In *Proceedings of the SIGDIAL 2010 Conference, The 11th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 55–58, 2010.

[63] Janet Hitzeman, Marc Moens, and Claire Grover. Algorithms for analysing the temporal structure of discourse. In *Proceedings of the seventh conference on European chapter of the Association for Computational Linguistics (EACL-95)*, pages 253–260, 1995.

[64] J. R. Hobbs. *Literature and cognition*. CSLI, Stanford, California, 1990. CSLI Lecture Notes.

[65] Jerry Hobbs, Mark Stickel, Douglas Appelt, and Paul Martin. Interpretation as abduction. *Artificial Intelligence*, 63:69–142, 1993.

[66] Jerry R. Hobbs. Resolving pronoun references. In Karen Sparck-Jones Barbara J. Grosz and Bonnie Lynn Webber, editors, *Readings in*

Natural Language Processing, pages 339–352. Morgan Kaufmann, Los Altos, 1978.

[67] Jerry R. Hobbs. Granularity. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence (IJCAI-85)*, pages 432–435, 1985.

[68] Jerry R. Hobbs, William Croft, Todd Davies, Douglas Edwards, and Kenneth Laws. Commonsense metaphysics and lexical semantics. *Computational Linguistics*, 13:241–250, July 1987.

[69] Eduard Hovy, Mitchell Marcus, Martha Palmer, Lance Ramshaw, and Ralph Weischedel. Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 57–60, 2006.

[70] Liang Huang and David Chiang. Forest rescoring: Faster decoding with integrated language models. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL-07)*, pages 144–151, 2007.

[71] Lucja Iwańska. Wayne state university: description of the uno natural language processing system as used for muc-6. In *Proceedings of the 6th conference on Message understanding (MUC-95)*, pages 263–277, 1995.

[72] Andrew Kehler. *Coherence, Reference and the Theory of Grammar*. CSLI, Stanford, California, 2002.

[73] Doo Soon Kim, Ken Barker, and Bruce Porter. Knowledge integration across multiple texts. In *Proceedings of the 5th International Conference on Knowledge Capture (KCAP-09)*, 2009.

[74] Doo Soon Kim, Ken Barker, and Bruce Porter. Improving the quality of text understanding by delaying ambiguity resolution. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING-10)*, pages 581 – 589, 2010.

[75] Doo Soon Kim and Bruce Porter. Handling granularity differences in knowledge integration. In *Proceedings of the Fall AAAI Symposium Series*, 2007.

[76] Doo Soon Kim and Bruce Porter. Integrating declarative knowledge: Issues, algorithms and future work. In *Proceedings of the Spring AAAI Symposium Series*, 2008.

[77] Doo Soon Kim and Bruce Porter. Kleo: A bootstrapping learning-by-reading system. In *Proceedings of the Spring AAAI Symposium Series*, 2009.

[78] Dan Klein and Christopher D. Manning. Parsing and hypergraphs. In *Proceedings of International Workshop on Parsing Technologies*, pages 123–134, 2001.

[79] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for*

Computational Linguistics (ACL-03), pages 423–430, 2003.

[80] Alistair Knott and Ted Sanders. The classification of coherence relations and their linguistic markers: An exploration of two languages. *Journal of Pragmatics*, 30(2):135–175, 1998.

[81] Henry Kucera and Nelson Francis. *Computational analysis of present-day American English*. Brown University Press, Providence, RI, 1967.

[82] Maria Lapata and Alex Lascarides. Learning sentence-internal temporal relations. *Journal of Artificial Intelligence Research*, 27:85–117, 2006.

[83] Alex Lascarides and Nicholas Asher. Temporal interpretation, discourse relations and common sense entailment. *Linguistics and Philosophy*, 16(5):437–493, 1993.

[84] Mark Lauer. Corpus statistics meet the noun compound: some empirical results. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics (ACL-95)*, pages 47–54, 1995.

[85] Douglas B. Lenat, R. V. Guha, Karen Pittman, Dexter Pratt, and Mary Shepherd. Cyc: Toward programs with common sense. *Communications of the ACM*, 33(8), 1990.

[86] Michael Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation (SIGDOC-06)*, pages 24–26, 1986.

[87] Beth Levin. *English Verb Classes and Alternations: a preliminary investigation*. University of Chicago Press, Chicago and London, 1993.

[88] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[89] Ziheng Lin, Min-Yen Kan, and Hwee Tou Ng. Recognizing implicit discourse relations in the penn discourse treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP-09)*, pages 343–351, 2009.

[90] Bill MacCartney and Christopher D. Manning. Natural logic for textual inference. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing (RTE-07)*, pages 193–200, 2007.

[91] Inderjeet Mani. A theory of granularity and its application to problems of polysemy and underspecification of meaning. In *In Principles of Knowledge Representation and Reasoning: Proceedings of the 6th International Conference (KR-98)*, pages 245–257, 1998.

[92] Inderjeet Mani, Marc Verhagen, Ben Wellner, Chong Min Lee, and James Pustejovsky. Machine learning of temporal relations. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-06)*, 2006.

[93] William C. Mann and Sandra A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–

281, 1988.

[94] Daniel Marcu and Abdessamad Echihabi. An unsupervised approach to recognizing discourse relations. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL-02)*, pages 368–375, 2002.

[95] Katja Markert and Malvina Nissim. Using the web for nominal anaphora resolution. In *EACL Workshop on the Computational Treatment of Anaphora*, pages 39–46, 2003.

[96] Marie-CatherinDe Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th Language Resources and Evaluation Conference (LREC-06)*, 2006.

[97] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. Semantic role labeling: an introduction to the special issue. *Computational Linguistics*, 34:145–159, 2008.

[98] John T. Maxwell III and Ronald M. Kaplan. A method for disjunctive constraint satisfaction. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 173–190. Kluwer Academic Publishers, Dordrecht, 1981.

[99] David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *Human Language Technologies: The*

2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-10), pages 28–36, 2010.

[100] Michael C. McCord. Slot Grammar: A system for simpler construction of practical natural language grammars. In R. Studer, editor, *Natural Language and Logic*, Lecture Notes in Computer Science, pages 118–145. Springer, New York, NY, 1990.

[101] Ivan V. Meza-Ruiz and Sebastian Riedel. Jointly identifying predicates, arguments and senses using markov logic. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-09)*, pages 155–163, 2009.

[102] Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technology (ACL-07)*, pages 192–199, 2008.

[103] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[104] Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. A novel use of statistical parsing to extract information from text. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference (NAACL-00)*, pages 226–233, 2000.

[105] Ruslan Mitkov, Branimir Boguraev, and Shalom Lappin. Introduction to the special issue on computational anaphora resolution. *Computational Linguistics*, 27(4):473–477, 2001.

[106] Kenneth S. Murray. KI: a tool for knowledge integration. In *Proceedings of the 13th national conference on Artificial intelligence (AAAI-96)*, pages 835–842, 1996.

[107] Srini Narayanan. Moving right along: A computational model of metaphoric reasoning about events. In *Proceedings of the National Conference on Artificial Intelligence (AAAI-99)*, pages 121–128. AAAI Press, 1999.

[108] Vincent Ng. Unsupervised models for coreference resolution. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 640–649, 2008.

[109] Joakim Nivre. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, 2003.

[110] Tomoko Ohta, Yuka Tateisi, and Jin-Dong Kim. The genia corpus: an annotated research abstract corpus in molecular biology domain. In *Proceedings of the second international conference on Human Language Technology Research (HLT-02)*, pages 82–86, 2002.

[111] Martha Palmer, Paul Kingsbury, and Daniel Gildea. The proposition

bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005.

[112] Siddharth Patwardhan, Satanjeev Banerjee, and Ted Pedersen. Senserelate::targetword: a generalized framework for word sense disambiguation. In *Proceedings of the ACL 2005 on Interactive poster and demonstration sessions*, pages 73–76, 2005.

[113] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan-Kaufmann, 1988.

[114] Thanh Phong Pham, Hwee Tou Ng, and Wee Sun Lee. Word sense disambiguation with semisupervised learning. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI-05)*, pages 1093–1098, 2005.

[115] Emily Pitler, Mridhula Raghupathy, Hena Mehta, Ani Nenkova, Alan Lee, and Aravind K. Joshi. Easily identifiable discourse relations. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pages 87–90, 2008.

[116] Massimo Poesio, Rahul Mehta, Axel Maroudas, and Janet Hitzeman. Learning to resolve bridging references. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 143–150, 2004.

[117] Massimo Poesio, Simone Ponzetto, and Yannick Versley. Computational models of anaphora resolution: a survey. Technical report, University of Trento, 2010.

[118] C. Pollard and I. A. Sag. *Head-driven Phrase Structure Grammar*. University of Chicago Press, Cambridge, MA, 1994.

[119] Rashmi Prasad, Nikhil Dinesh, Alan Lee, Eleni Miltsakaki, Livio Robaldo, Aravind K. Joshi, and Bonnie L. Webber. The penn discourse treebank 2.0. In *Proceedings of the 6th Language Resources and Evaluation Conference (LREC-08)*, 2008.

[120] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The necessity of syntactic parsing for semantic role labeling. In *Proceedings of the 19th international joint conference on Artificial intelligence (IJCAI-05)*, pages 1117–1123, 2005.

[121] Vasin Punyakanok, Dan Roth, and Wen-tau Yih. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34:257–287, June 2008.

[122] Michaela Regneri, Markus Egg, and Alexander Koller. Efficient processing of underspecified discourse representations. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies (HLT-08)*, pages 245–248, 2008.

[123] Uwe Reyle. Underspecified discourse representation structures and their logic. *Logic Journal of the IGPL*, 3(2-3):473–488, 1995.

[124] Matthew Richardson and Pedro Domingos. *Markov logic networks*. Kluwer Academic Publishers, 2006.

[125] Stefan Riezler, Tracy H. King, Ronald M. Kaplan, Richard S. Crouch, John T. Maxwell III, and Mark Johnson. Parsing the Wall Street Journal using a lexical-functional grammar and discriminative estimation techniques. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL-02)*, pages 271–278, 2002.

[126] Roger C. Schank. *A Conceptual Dependency Representation for a Computer-Oriented Semantics*. PhD thesis, University of Texas, Austin, 1969.

[127] Lenhart Schubert. Can we derive general world knowledge from texts? In *Proceedings of the second international conference on Human Language Technology Research (HLT-02)*, pages 94–97, 2002.

[128] Lenhart Schubert and Chung Hee Hwang. Episodic logic meets little red riding hood: a comprehensive natural representation for language understanding. In *Natural language processing and knowledge representation*, pages 111–174. 2000.

[129] Karin Kipper Schuler. *Verbnet: a broad-coverage, comprehensive verb lexicon*. PhD thesis, University of Pennsylvania, 2005.

[130] Push Singh, Thomas Lin, Erik T. Mueller, Grace Lim, Travell Perkins, and Wan Li Zhu. Open mind common sense: Knowledge acquisition from the general public. *Lecture Notes in Computer Science*, 2519:1223–1237, 2002.

[131] Barry Smyth and Mark T. Keane. Adaptation-guided retrieval: Questioning the similarity assumption in reasoning. *Artificial Intelligence*, 102:249–293, 1998.

[132] Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Semantic taxonomy induction from heterogeneous evidence. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL-06)*, 2006.

[133] Wee Meng Soon, Hwee Tou Ng, and Chung Yong Lim. A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics*, 27(4):521–544, 2001.

[134] John Sowa. *Conceptual Structures*. Addison-Wesley, 1984.

[135] Michael Strube and Udo Hahn. Functional centering - grounding referential coherence in information structure. *Computational Linguistics*, 25(3):309–344, 1999.

[136] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of the Twelfth Conference on*

Computational Natural Language Learning (CoNLL-08), pages 159–177, 2008.

[137] Charles Sutton and Andrew McCallum. Joint parsing and semantic role labeling. In *Proceedings of the 9th Conference on Computational Natural Language Learning (CONLL-05)*, pages 225–228, 2005.

[138] Matthew E. Taylor, Cynthia Matuszek, Bryan Klimt, and Michael Witbrock. Autonomous classification of knowledge into an ontology. In *Proceedings of the 20th International FLAIRS Conference (FLAIRS-07)*, 2007.

[139] Joel R. Tetreault. Analysis of syntax-based pronoun resolution methods. In *Proceedings of the 27th Annual Meeting of the Association for Computational Linguistics (ACL-99)*, pages 602–605, 1999.

[140] Masaru Tomita. *Efficient Parsing for Natural Language — A Fast Algorithm for Practical Systems*. International Series in Engineering and Computer Science. Kluwer, Hingham, MA, 1986.

[141] Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation (SEMEVAL-10)*, pages 57–62, 2010.

[142] Renata Viera and Massimo Poesio. An empirically based system for

processing definite descriptions. *Computational Linguistics*, 26(2):539–593, 2000.

[143] Marc Vilain, John Burger, John Aberdeen, Dennis Connolly, and Lynette Hirschman. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding (MUC-95)*, pages 45–52, 1995.

[144] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings 17th International Conference on Machine Learning (ICML-00)*, pages 1103–1110, 2000.

[145] Ben Wellner, Andrew McCallum, Fuchun Peng, and Michael Hay. An integrated, conditional model of information extraction and coreference with application to citation matching. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence (UAI-04)*, pages 593–601, 2004.

[146] Yorick A. Wilks, Brian M. Slator, and Louise M. Guthrie. *Electric words: dictionaries, computers, and meanings*. MIT Press, Cambridge, MA, 1996.

[147] Y. Y. Yao. Granular computing: basic issues and possible solutions. In *Proceedings of the 5th Joint Conference on Information Sciences*, pages 186–189, 2000.

[148] David Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33th Annual Meeting of the Association for Computational Linguistics (ACL-95)*, pages 189–196, 1995.

[149] Peter Yeh, Bruce Porter, and Ken Barker. Mining transformation rules for semantic matching. In *ECML/PKDD 2nd International Workshop on Mining Graphs, Trees, and Sequences*, pages 83–94, 2004.

[150] Peter Yeh, Bruce Porter, and Ken Barker. A unified knowledge based approach for sense disambiguation and semantic role labeling. In *Proceedings of 20th National Conference on Artificial Intelligence (AAAI-06)*, 2006.

[151] Peter Z. Yeh, Bruce Porter, and Ken Barker. Using transformations to improve semantic matching. In *Proceedings of the 2nd International Conference on Knowledge Capture (KCAP-03)*, pages 180–189, 2003.

[152] Peter Z. Yeh, Colin A. Puri, and Alex Kass. A knowledge based approach for capturing rich semantic representations from text for intelligent systems. *International Journal of Advanced Intelligence Paradigms*, 2:33–48, November 2010.

[153] L. A. Zadeh. Fuzzy logic = computing with words. *IEEE Transactions on Fuzzy Systems*, 4(2):103–111, 1996.

[154] Wlodek Zadrozny and Karen Jensen. Semantics of paragraphs. *Computational Linguistics*, 17:171–209, June 1991.

[155] Zhi Zhong, Hwee Tou Ng, and Yee Seng Chan. Word sense disambiguation using ontonotes: an empirical study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-08)*, pages 1002–1010, 2008.

Index

Abstract, vi
Acknowledgments, v
Appendices, 161

Bibliography, 191

Dedication, iv

Vita

Doo Soon Kim was born in Seoul, South Korea on 12th April 1978, to Dr. Sun Yo Kim and Jung Sook An. He graduated from Hansung Science High School in 1996, after which he attended the Korean Advanced Institute of Science and Technology (KAIST) and received the Bachelor of Science in Computer Sciences. Upon graduation, and as one of the top four students in the entire graduating class, he was awarded the KAIST Action Committee prize.

Following graduation, Doo Soon worked for a software company, OCI Communication, to fulfill his military duty. He then studied at the University of Texas at Austin from 2004 to 2011 to pursue his PhD in the Computer Sciences program. During his tenure at the University, he received the Samsung Lee Kun Hee Scholarship, from 2004 to 2008. His research primarily focuses on Artificial Intelligence, Knowledge Representation and Reasoning, and Natural Language Understanding.

Permanent address: 12370 Alameda Trace Circle Apt. 1312
Austin, Texas 78727

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.